a little systemtap

detailyang@gmail.com

2016 11-24

什么是 systemtap

Linux 下万能的 Trace 工具。

Linux 下目前的 Tracer

systemtap 能做什么

跟踪系统调用

stap -e 'probe syscall.open {printf("%s(%d) is opening %s\n", execname(), pid(), user_string($filename))}'

vminfo(710) is opening /var/run/utmp
less(11933) is opening /root/.lesshst
cat(12198) is opening /etc/ld.so.cache
cat(12198) is opening /lib64/libc.so.6
cat(12198) is opening /usr/lib/locale/locale-archive
cat(12198) is opening /tmp/123

跟踪内核态

```
probe kernel.function("tcp_retransmit_skb") {
    rto = tcp_get_info_rto($sk)
    saddr   = format_ipaddr(__ip_sock_saddr($sk), __ip_sock_family($sk))
    daddr   = format_ipaddr(__ip_sock_daddr($sk), __ip_sock_family($sk))
    sport   = __tcp_sock_sport($sk)
    dport   = __tcp_sock_dport($sk)
    lastrto = record[saddr, sport, daddr, dport]
    state = tcp_ts_get_info_state($sk)

    if (lastrto != rto) {
        if (lastrto) {
            printf("%s:%d => %s:%d STATE:%s RTO:%d -> %d (ms)\n", saddr, sport,
                daddr, dport, tcp_sockstate_str(state), lastrto/1000, rto/1000)
        } else {
            printf("%s:%d => %s:%d STATE:%s RTO:%d (ms)\n", saddr, sport,
                daddr, dport, tcp_sockstate_str(state), rto/1000)
        }
    }

    record[saddr, sport, daddr, dport] = rto
}
```

跟踪应用程序

```
probe process(“/usr/bin/redis-server”).function(“getConmmand") {
        printf(“id:%d %s %s\n", $c->id, user_string($c->argv[0]-
        >ptr), user_string($c->argv[1]->ptr))
}
```

id:2 get abcd
id:2 get a

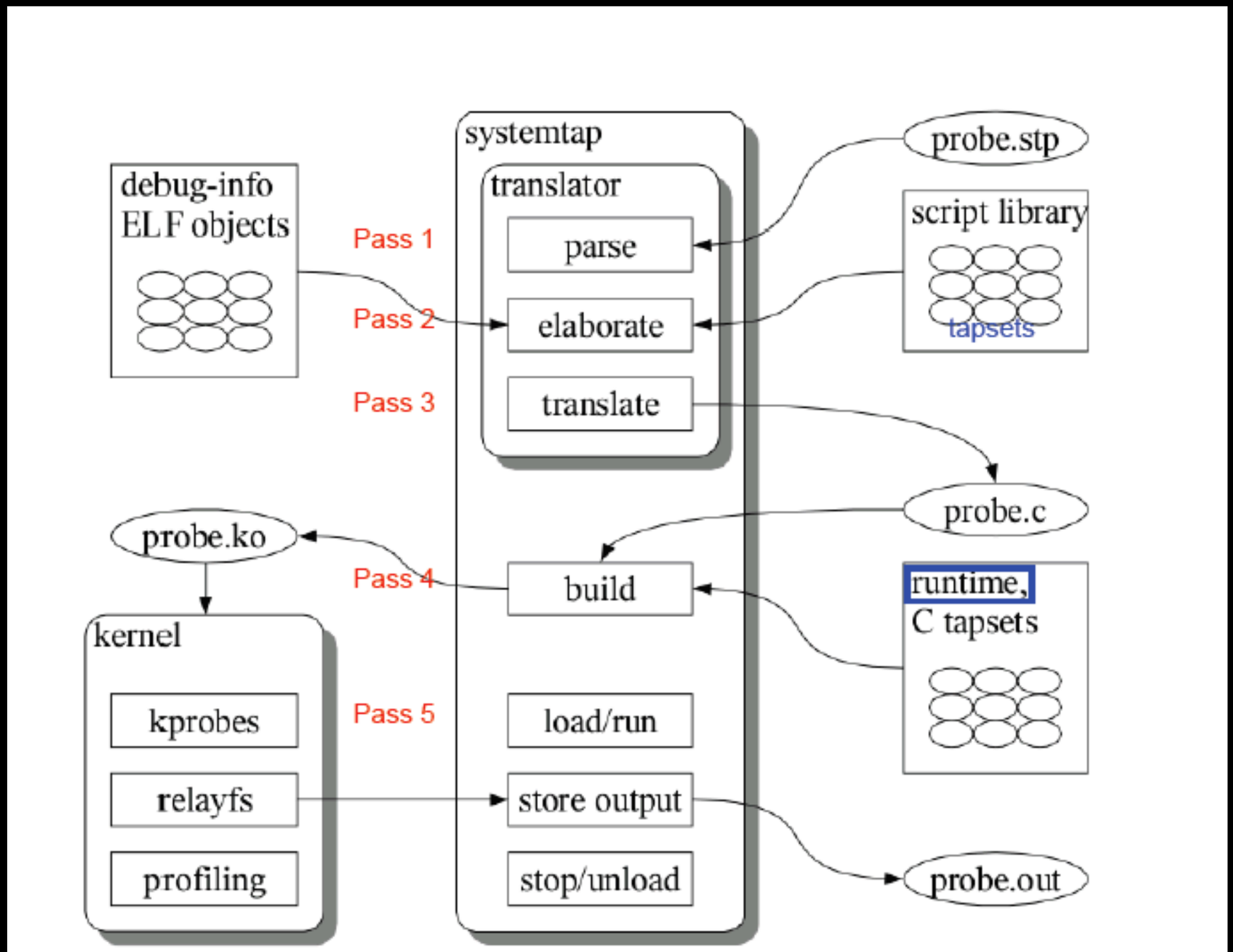
systemtap 安装

yum install -y systemtap

早先的内核版本依赖utrace补丁

3.50 使用内核 uprobes 和 uretprobes

systemtap 原理



调试符号是什么

Linux 目前的调试格式 dwarf

调试符号是什么

源代码中每个对象的名称。例如变量、函数、类型等，它们都有一个名称，以及其它的相关信息：变量有类型、地址等信息；函数有返回值类型、参数类型、地址等信息；类型有长度等信息。编译器在编译每个源文件的时候都会收集该源文件中的符号的信息，在生成目标文件的时候将这些信息保存到符号表中。链接器使用符号表中的信息将各个目标文件链接成可执行文件，同时将多个符号表整合成一个文件，这个文件就是用于调试的符号文件，它既可以嵌入可执行文件中，也可以独立存在

```
stap -L 'process("/usr/bin/redis-
server").function("zzlPrev")'
```

```
process("/usr/bin/redis-server").function("zzlPrev@/usr/
src/debug/redis-2.8.19/src/t_zset.c:720") $zl:unsigned
  char* $eptr:unsigned char** $sptr:unsigned char**
```

分离的 debug info

因为程序带上调试符号会非常大，而且不是所有的用户需要调试符号。

所以有了分离的 debug info。

RPM的机制

/usr/lib/rpm/find-debuginfo.sh --strict-build-id xx
extracting debug info from xx

RPM的机制

objcopy xx xx.debug
strip --strip-debug ./xx

奥秘在于二进制中的 build-id

RPM的机制

objcopy xx xx.debug
strip --strip-debug ./xx

奥秘在于程序中的 build-id

objdump -s -j .note.gnu.build-id /usr/bin/redis-server

RPM的机制

```
[root@localhost ~]# objdump -s -j .note.gnu.build-id /usr/bin/redis-server

/usr/bin/redis-server:     file format elf64-x86-64

Contents of section .note.gnu.build-id:
 0274 04000000 14000000 03000000 474e5500  ............GNU.
 0284 13f3476a 9a652bc2 bb96eba1 d0e1d89c  ..Gj.e+.........
 0294 570d3966                             W.9f
```

build-id =
0x13f3f3476a9a652bc2bb96eba1d0e1d89c570d3966

RPM的机制

0x13　　f3f3476a9a652bc2bb96eba1d0e1d89c570d3966

ls /usr/lib/debug/.build-id/13/

f3476a9a652bc2bb96eba1d0e1d89c570d3966

ngx-req-watch

```
[root@localhost systemtap-toolkit]# ./ngx-req-watch -p 5614
WARNING: watching /opt/nginx/sbin/nginx(8521 8522 8523 8524) requests
nginx(8523) GET URI:/123?a=123 HOST:127.0.0.1 STATUS:200 FROM 127.0.0.1 FD:16 RT:
0ms
nginx(8523) GET URI:/123?a=123 HOST:127.0.0.1 STATUS:200 FROM 127.0.0.1 FD:16 RT:
0ms
```

libcurl-watch-req

```
[root@localhost systemtap-toolkit]# ./libcurl-watch-req
WARNING: Tracing libcurl (0) ...
curl(23759) URL:http://www.google.com RT:448(ms) RTCODE:0
curl(23767) URL:http://www.facebook.com/asdfasdf RT:596(ms) RTCODE:0
curl(23769) URL:https://www.facebook.com/asdfasdf RT:902(ms) RTCODE:0
```

redis-watch-req

```
[root@localhost systemtap-toolkit]# ./redis-watch-req -p 23261
WARNING: watching /usr/bin/redis-server(23261) requests
redis-server(23261) RT:30(us) REQ: id:2 fd:5 ==> get a #-1 RES: #9
redis-server(23261) RT:23(us) REQ: id:2 fd:5 ==> set a #12 RES: #5
```

pdomysql-watch-query

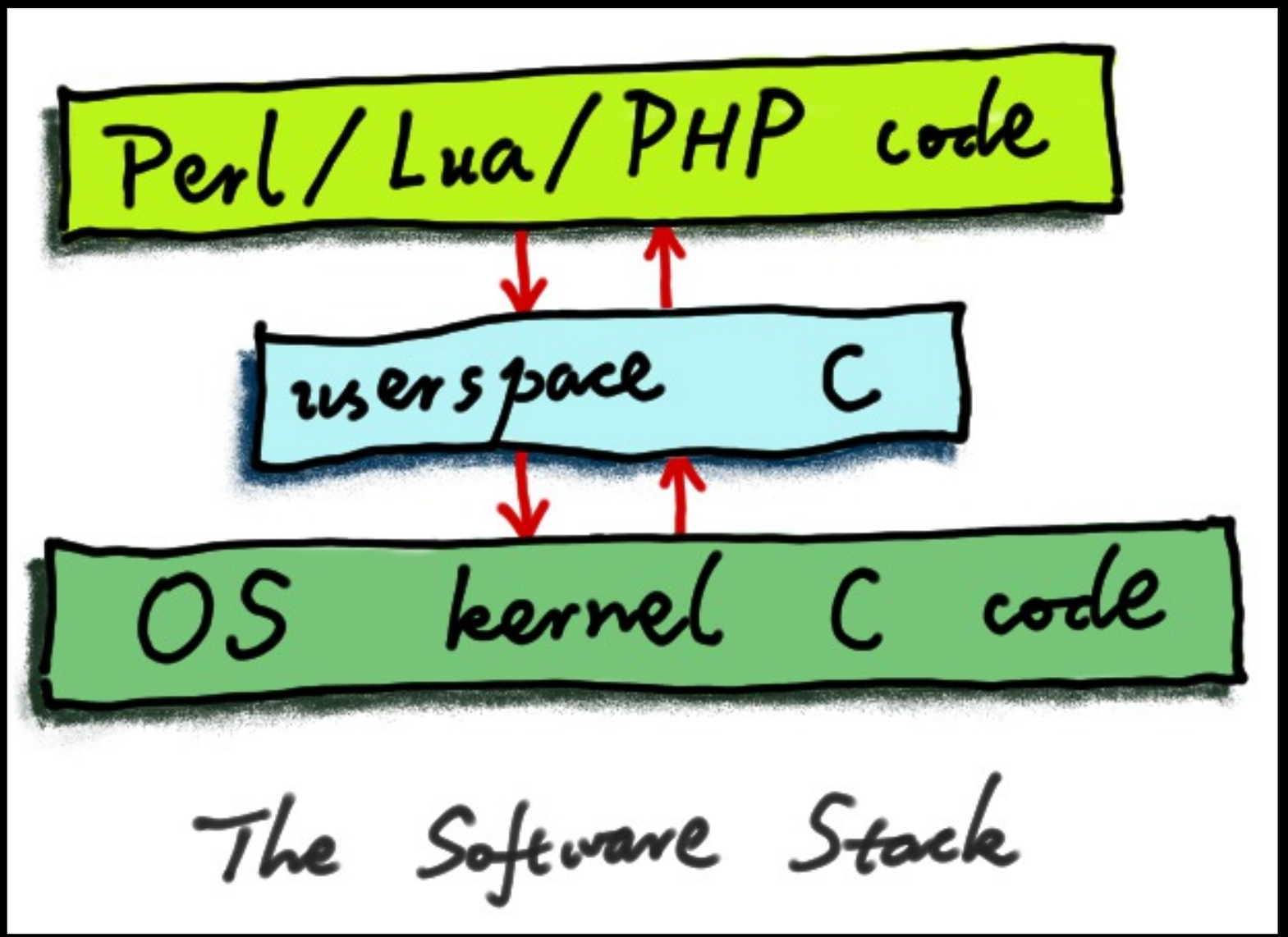
[root@localhost systemtap-toolkit]# ./pdomysql-watch-query -l /usr/lib64/php/modules/pdo_mysql.so

Tracing pdo-mysql (0)
php-fpm(12896) 172.17.10.196:3306@root: SELECT * from person RT:0(ms) RTCODE:1
php-fpm(12896) 172.17.10.196:3306@root: SELECT * from person RT:8(ms) RTCODE:1
php-fpm(12896)172.17.10.196:3306@root: SELECT sleep(5) RT:5012(ms) RTCODE:1

TCP-Retrans

```
[root@localhost systemtap-toolkit]# ./tcp-retrans
Printing tcp retransmission
10.0.2.15:49896 -> 172.17.9.41:80 state:TCP_SYN_SENT rto:0 -> 1000 ms
10.0.2.15:49896 -> 172.17.9.41:80 state:TCP_SYN_SENT rto:1000 -> 2000 ms
10.0.2.15:49896 -> 172.17.9.41:80 state:TCP_SYN_SENT rto:2000 -> 4000 ms
```

beyond c land



MySQL DTRACE

Group	Probes
Connection	connection-start, connection-done
Command	command-start, command-done
Query	query-start, query-done
Query Parsing	query-parse-start, query-parse-done
Query Cache	query-cache-hit, query-cache-miss
Query Execution	query-exec-start, query-exec-done
Row Level	insert-row-start, insert-row-done
	update-row-start, update-row-done
	delete-row-start, delete-row-done
Row Reads	read-row-start, read-row-done
Index Reads	index-read-row-start, index-read-row-done
Lock	handler-rdlock-start, handler-rdlock-done
	handler-wrlock-start, handler-wrlock-done
	handler-unlock-start, handler-unlock-done
Filesort	filesort-start, filesort-done

Python DTRACE

```
$ python3.6 -q &
$ sudo dtrace -l -P python$!  # or: dtrace -l -m python3.6

   ID   PROVIDER            MODULE                          FUNCTION NAME
29564 python18035         python3.6         _PyEval_EvalFrameDefault function-entry
29565 python18035         python3.6            dtrace_function_entry function-entry
29566 python18035         python3.6         _PyEval_EvalFrameDefault function-return
29567 python18035         python3.6           dtrace_function_return function-return
29568 python18035         python3.6                          collect gc-done
29569 python18035         python3.6                          collect gc-start
29570 python18035         python3.6         _PyEval_EvalFrameDefault line
29571 python18035         python3.6                maybe_dtrace_line line
```

Java DTRACE

```
provider hotspot {
  probe vm-init-begin();
  probe vm-init-end();
  probe vm-shutdown();
  probe class-loaded(
      char* class_name, uintptr_t class_name_len, uintptr_t class_loader_id, bool is_shared);
  probe class-unloaded(
      char* class_name, uintptr_t class_name_len, uintptr_t class_loader_id, bool is_shared);
  probe gc-begin(bool is_full);
  probe gc-end();
  probe mem-pool-gc-begin(
      char* mgr_name, uintptr_t mgr_name_len, char* pool_name, uintptr_t pool_name_len,
      uintptr_t initial_size, uintptr_t used, uintptr_t committed, uintptr_t max_size);
  probe mem-pool-gc-end(
      char* mgr_name, uintptr_t mgr_name_len, char* pool_name, uintptr_t pool_name_len,
      uintptr_t initial_size, uintptr_t used, uintptr_t committed, uintptr_t max_size);
  probe thread-start(
      char* thread_name, uintptr_t thread_name_length,
      uintptr_t java_thread_id, uintptr_t native_thread_id, bool is_daemon);
  probe thread-stop(
      char* thread_name, uintptr_t thread_name_length,
      uintptr_t java_thread_id, uintptr_t native_thread_id, bool is_daemon);
  probe method-compile-begin(
      char* class_name, uintptr_t class_name_len,
      char* method_name, uintptr_t method_name_len,
      char* signature, uintptr_t signature_len);
  probe method-compile-end(
      char* class_name, uintptr_t class_name_len,
      char* method_name, uintptr_t method_name_len,
      char* signature, uintptr_t signature_len,
      bool is_success);
```

Q&A