Table of Contents

Using BlockTrace and SystemTap

To instrument the I/O and visualize it, I've put together a collection of scripts I call BlockTrace, which depend on SystemTap.

Because we are inserting code into the Linux kernel, you need root privileges; all the systemtap scripts must be run as root.

SystemTap Installation

I assume you are using RHEL5. Do the following:

- `yum install yum-utils elfutils-devel`. A few background dependencies
- `debuginfo-install kernel`. This pulls in the debug symbols for your kernel. Depending on your kernel version, and what's in the upstream CentOS / SLC repository, this might not work. On many of the nodes I use, I've found this repository from Oracle http://oss.oracle.com/el5/debuginfo/ to contain all the RPMs I needed
- `yum install git`. You'll need to have the EPEL repository enabled and installed on your machine. If you can't do this easily, install git from source.
- Follow the systemtap install instructions.

I've installed it into /usr for myself. You should have the `stap` binary in your environment.

Run an Example

SystemTap installs quite a few examples. Let's run the `iotop` example; this is a re-implementation of iotop in under 30 lines of scripting.

```
[root@node149 ~]# stap /usr/share/doc/systemtap/examples/io/iotop.stp
         Process      KB Read   KB Written
          stapio         3216            0
           gmond           32            0
      irqbalance           16            0
           snmpd           10            0
           klogd            3            0

         Process      KB Read   KB Written
          stapio         3328            0
            sshd           16            0
           gmond           16            0
           snmpd           10            0
```

If you can run this, then we're set to move on. If you see some errors talking about debug info for your kernel, revisit the previous step. Make sure the kernel debuginfo RPM you've installed matches the output of "uname -r" **exactly**.

Check out and run BlockTrace

The BlockTrace collection of scripts is kept in UNL subversion:

```
svn co svn://t2.unl.edu/brian/BlockTrace
```

The SystemTap script is called `blk_trace.stp`. You should be able to execute it directly. It takes two parameters:

1. Process name. Usually, I set this to "cmsRun" if I'm testing CMSSW. In the examples below, we'll start with something simpler.
2. File name. Provide the file name as passed to open(). If you are using the framework, use the 'file' protocol. Anything specified after 'file:' is what you put here. So, for example, if your python configuration says the input is '=file:///foo/bar.root=', then the file name you want is '=///foo/bar.root='. Make sure you keep all the slashes.

The blk_trace.stp records three things:

1. System calls.
2. Queuing and completion of block-level requests (BIO.*).
3. Queueing and completion of block requests sent to the device driver (RQ.*)

For an example run, open up two terminals. In the first terminal, execute `blk_trace.stp` (with the `-v` and `-g` options):

```
./blk_trace.stp -v -g cat /tmp/foo.txt
```

So, it will monitor all activity from a process named `cat` on the file `/tmp/foo.txt` (make sure foo.txt exists on your system; if not, create it and throw a few tens of KB into it).

In the other terminal, drop all the file system cache information (otherwise your file might only be served by memory, making this a very boring experience):

```
echo 1 > /proc/sys/vm/drop_caches
```

Now, cat out the contents of `/tmp/foo.txt`:

```
cat /tmp/foo.txt > /dev/null
```

You should see SystemTap output like the following:

```
[root@node149 BlockTrace]# ./blk_trace.stp -v -g cat /tmp/foo.txt
Pass 1: parsed user script and 65 library script(s) using 75204virt/19724res/1676shr kb, in 140us
Pass 2: analyzed script: 18 probe(s), 26 function(s), 19 embed(s), 20 global(s) using 151464virt/
Pass 3: translated to C into "/tmp/stapWyYxEB/stap_9c8823afc91cb7dea83b4797196732cf_36379.c" usin
Pass 4: compiled C into "stap_9c8823afc91cb7dea83b4797196732cf_36379.ko" in 3250usr/300sys/4707re
Pass 5: starting run.
cat( 5531) sys_open "/tmp/foo.txt", O_RDONLY elapsed_us=0
cat( 5531) cmd=sys_read fd=3 count=4096 read_idx=0 elapsed_us=6
cat( 5531) cmd=BIO.start inode=2490378 fd=3 sector=20086784 phys_segs=1 size=4096 flags=BIO_UPTOD
cat( 5531) cmd=BIO.start inode=2490378 fd=3 sector=25174032 phys_segs=3 size=12288 flags=BIO_UPTO
cat( 5531) cmd=RQ.start fd=3 sector=210146826 phys_segs=3 nr_sectors=24 elapsed_us=76
cat( 5531) cmd=RQ.start fd=3 sector=205059578 phys_segs=1 nr_sectors=8 elapsed_us=125
swapper(    0) cmd=RQ.return sector=205059578 phys_segs=1 nr_sectors=8 calltime_us=5912 elapsed_u
swapper(    0) cmd=BIO.return inode=2490378 sector=20086784 phys_segs=1 size=4096 calltime_us=602
cat( 5531) cmd=sys_read.return retval=4096 read_idx=0 calltime_us=6075 elapsed_us=6081
cat( 5531) cmd=sys_read fd=3 count=4096 read_idx=1 elapsed_us=6094
cat( 5531) cmd=BIO.start inode=2490378 fd=3 sector=25174056 phys_segs=3 size=12288 flags=BIO_UPTO
cat( 5531) cmd=BIO.start inode=2490378 fd=3 sector=26484760 phys_segs=1 size=4096 flags=BIO_UPTOD
cat( 5531) cmd=BIO.start inode=2490378 fd=3 sector=26484768 phys_segs=1 size=4096 flags=BIO_UPTOD
block_bio_backmerge
cat( 5531) cmd=BIO.start inode=2490378 fd=3 sector=26484776 phys_segs=1 size=4096 flags=BIO_UPTOD
block_bio_backmerge
cat( 5531) cmd=BIO.start inode=2490378 fd=3 sector=26484784 phys_segs=1 size=4096 flags=BIO_UPTOD
block_bio_backmerge
cat( 5531) cmd=BIO.start inode=2490378 fd=3 sector=26484792 phys_segs=1 size=4096 flags=BIO_UPTOD
block_bio_backmerge
cat( 5531) cmd=RQ.start fd=3 sector=210146850 phys_segs=3 nr_sectors=24 elapsed_us=6176
cat( 5531) cmd=RQ.start fd=3 sector=211457554 phys_segs=5 nr_sectors=40 elapsed_us=6192
```

```
swapper(    0) cmd=RQ.return sector=210146826 phys_segs=3 nr_sectors=24 calltime_us=7855 elapsed_
swapper(    0) cmd=BIO.return inode=2490378 sector=25174032 phys_segs=3 size=12288 calltime_us=78
cat( 5531) cmd=sys_read.return retval=4096 read_idx=1 calltime_us=1857 elapsed_us=7951
cat( 5531) cmd=sys_read fd=3 count=4096 read_idx=2 elapsed_us=7957
cat( 5531) cmd=sys_read.return retval=4096 read_idx=2 calltime_us=7 elapsed_us=7964
cat( 5531) cmd=sys_read fd=3 count=4096 read_idx=3 elapsed_us=7969
cat( 5531) cmd=sys_read.return retval=4096 read_idx=3 calltime_us=6 elapsed_us=7975
cat( 5531) cmd=sys_read fd=3 count=4096 read_idx=4 elapsed_us=7979
cat( 5531) cmd=BIO.start inode=0 fd=3 sector=26532584 phys_segs=0 size=4096 flags=BIO_UPTODATE el
cat( 5531) cmd=RQ.start fd=3 sector=211505378 phys_segs=1 nr_sectors=8 elapsed_us=8007
swapper(    0) cmd=RQ.return sector=210146850 phys_segs=3 nr_sectors=24 calltime_us=2451 elapsed_
swapper(    0) cmd=BIO.return inode=2490378 sector=25174056 phys_segs=3 size=12288 calltime_us=25
swapper(    0) cmd=RQ.return sector=211457554 phys_segs=5 nr_sectors=40 calltime_us=4980 elapsed_
swapper(    0) cmd=BIO.return inode=2490378 sector=26484760 phys_segs=1 size=4096 calltime_us=504
swapper(    0) cmd=BIO.return inode=2490378 sector=26484768 phys_segs=1 size=4096 calltime_us=504
swapper(    0) cmd=BIO.return inode=2490378 sector=26484776 phys_segs=1 size=4096 calltime_us=504
swapper(    0) cmd=BIO.return inode=2490378 sector=26484784 phys_segs=1 size=4096 calltime_us=503
swapper(    0) cmd=BIO.return inode=2490378 sector=26484792 phys_segs=1 size=4096 calltime_us=504
swapper(    0) cmd=RQ.return sector=211505378 phys_segs=1 nr_sectors=8 calltime_us=5727 elapsed_u
cat( 5531) cmd=BIO.start inode=2490378 fd=3 sector=26532592 phys_segs=8 size=32768 flags=BIO_UPTO
cat( 5531) cmd=BIO.start inode=2490378 fd=3 sector=26534088 phys_segs=3 size=12288 flags=BIO_UPTO
cat( 5531) cmd=BIO.start inode=2490378 fd=3 sector=26534632 phys_segs=5 size=20480 flags=BIO_UPTO
cat( 5531) cmd=sys_read.return retval=4096 read_idx=4 calltime_us=5803 elapsed_us=13782
cat( 5531) cmd=sys_read fd=3 count=4096 read_idx=5 elapsed_us=13791
cat( 5531) cmd=sys_read.return retval=4096 read_idx=5 calltime_us=7 elapsed_us=13798

....

cat( 5531) cmd=sys_read fd=3 count=4096 read_idx=256 elapsed_us=70940
cat( 5531) cmd=sys_read.return retval=0 read_idx=256 calltime_us=13 elapsed_us=70953
cat(5531) sys_close filename=/tmp/foo.txt elapsed_us=70959
```

This is a detailed log of syscall, block, and driver activity. It's a lot of output, so we'll discuss next how to visualize it.

This topic: Main > CmsIoBlockTrace
Topic revision: r2 - 2010-05-13 - BrianBockelman