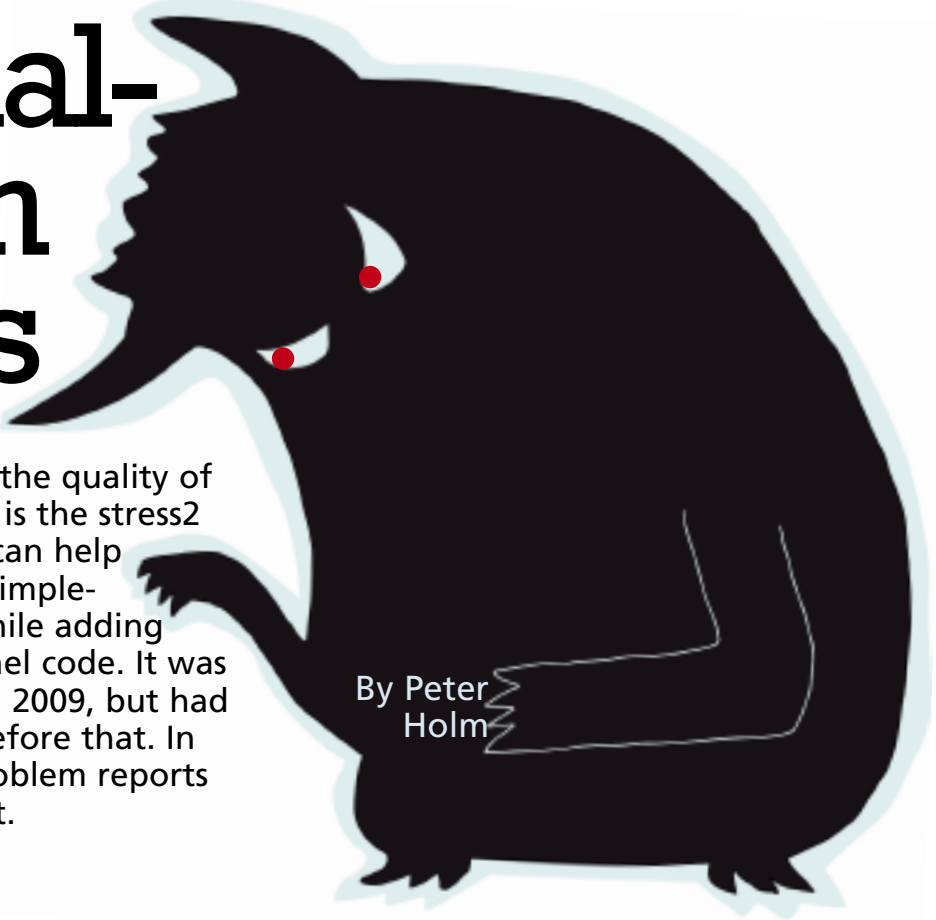


USING FUZZY TESTING TO BUILD

Industrial-Strength Systems

One tool to improve the quality of the FreeBSD kernel is the stress2 stress-test suite, which can help you expose design and implementation problems while adding to or updating the kernel code. It was first checked into svn in 2009, but had been in use for years before that. In 2014 more than 100 problem reports were generated using it.



By Peter Holm

How I Got Started with Stress Testing

Sometime during the nineties I was asked by the support group in the company I worked for to look at a customer's computer that would crash every time it ran a vendor database application. To my amazement, the computer panicked in a frequently used syscall. Stepping through the code, I realized that it was a simple stack-trashing bug in a user-mode program that had triggered the problem. Two things were forever imprinted on my mind: a) although it was interesting to meet the customer's top management, it would be much nicer to find these kinds of errors back at the office, and b) I was amazed that a common user-mode bug had crashed the computer! The same day, I wrote my first stress test program—syscall fuzzer—and that kept a lot of people busy for days.

Test Design

A widely used test in the FreeBSD Project is "buildworld." It is a good test, but when you have run it a few times in a row there is no gain in continuing. Stress2 uses a different strategy for most of the test cases, as tests run differently each time they are run—different runtime, different number of threads, and different VM pressure. The benefit of this strategy is that it provides greater coverage. The downside is that it is slightly harder to reproduce errors, but this problem is really negligible compared to the benefits.

Running the same test in multiple threads is used by many scenarios to stress locking of shared resources. Some tests synchronize the threads, but others will not, in the hope of achieving a wider test coverage. A cornerstone of Stress2 is that VM pressure is added to most tests to trigger waiting points in the kernel. A large number of tests have run without any detectable issues, but the moment VM pressure is added, panics and deadlocks show up.

A few tests rely on fuzzing. An example is the mmap(2) family of tests.

A Typical Work Day with Stress2

I get a patch from one of the FreeBSD committers that is supposed to fix a panic seen. The problem may already be triggered by some of the existing test scenarios or a new test has to be written. Once the problem can be repeated within a reasonable time frame, it's time to test the patch. When the fix has been verified, I typically run all of the other scenarios, just in case the fix has some side effect, which happens surprisingly often. It is not unusual for a patch to go through a few revisions before commit. On a really productive day, I'm able to go through three or four revisions. I try to generate bug reports with enough information to ensure a fast patch update. A bug report typically looks like this:

<https://people.freebsd.org/~pho/stress/log/kostik833.txt>.

A full test usually takes more than 24 hours before I'm confident the fix is OK.

When working on a new test scenario, the by-product is quite often one or more new test cases that trigger other problems than the one reported. These are marked for commit, even though they may be WIP. The philosophy here is that any test that can crash the kernel is a good test.

As can be seen by this example, it is a team effort. I work with a bunch of extremely talented people, and as a team we are incredibly productive.

Memory Leaks

Memory leaks get introduced on a regular basis. I check this by watching `vmstat -m` and `-z`. There is a script I run during tests that does this automatically: `stress2/tools/vmstat.sh`

Test Hardware

I have observed that different types of hardware make a difference, as some problems can only be reproduced on one type. Disk speed also plays a role and so does the number of CPUs. Lately I was reminded of the importance of this. I had a patch for evaluation, which survived all tests. Others, however, found that a continuous `-j7 buildworld` on a 6-core machine configured with 1GB of RAM would trigger a panic. Having swap configured is mandatory for quite a few of the tests. Without swap, it is very hard to balance the right amount of pressure. That is, without a swap disk, too much VM pressure triggers OOM killing. I test both `i386` and `amd64` on real hardware and I use a serial console for all test hosts, so I can log the output.

Examples

The following describe a few test scenarios I found especially interesting. All are found in the `stress2/misc` directory. Common for all tests in this directory is that they are all complete and self-contained test scenarios, implemented as shell scripts. The majority are regression tests; that is, they have once triggered a panic or a deadlock. The tests can be run individually. Most, but not all, tests use a memory-based file system for the tests. The primary benefit is that file system is in an initial consistent state.

The following show what a test in `stress2/misc` can look like:

```
#!/bin/sh

# "panic: not suspended thread 0xc674c870" seen.

for i in `find /proc ! -type d`; do
    dd if=$i of=/dev/null > /dev/null 2>&1
    dd if=/dev/random of=$i > /dev/null 2>&1
done
```

Many of the tests in `stress2/misc` use the more general stress test programs found in `stress2/testcases`.

trim6.sh

Quite often test cases written for one purpose also catch different problems. This test case was originally written for a problem deleting a large file on a file system with option TRIM enabled, but was later able to trigger both a deadlock and a panic during file creation. The test in this example is quite

simple: write a very large file to a fast (SSD) disk. From the r287361 commit log: By doing file extension fast, it is possible to create excess supply of the D_NEWBLK kinds of dependencies (i.e., D_ALLOCDIRECT and D_ALLOCINDIR) which can exhaust kmem.

crossmp.sh

A different and productive example of parallelization is the Cross Mount Point test case, which is responsible for 6 bug reports. The tests mount and unmount 15 different file systems / mount points in parallel.

callout_reset_on.sh

The scenario I have spent the most time on is from pr. kern/166340, a very detailed bug report: Processes under FreeBSD 9.0 would hang in uninterruptible sleep with apparently no syscall (empty wchan). A later change to the callout wheel would trigger a panic: Bad link elm 0xffffffff80012ba8ec8 prev->next != elm with this scenario.

mmap10.sh

This is one of the fuzz test cases. The general idea is to pass random values to system calls in an attempt to flush out errors in the code. Once you get past the simple missing parameter validation problems, more interesting problems tend to surface. This test scenario generated the following unique problems by passing random values to mlock(2), mprotect(2), and mlockall(2):

```
panic: deadlkres: possible deadlock detected for 0xcb0ea930, blocked for 1801709 ticks
panic: pmap active 0xffffffff800a90cfd78
panic: vm_fault_copy_wired: page missing
panic: vm_object_backing_scan: object mismatch
panic: vm_page_dirty: page is invalid!
panic: vmSPACE_fork: entry 0xffffffff80019793d00 eflags 50c
```

rename3.sh

Some test scenarios are written or suggested by others. For example, this small rename test scenario by Tor Egge: "Test vulnerability to transient failures when a directory closer to the root directory is renamed". This has triggered multiple deadlocks.

isofs2.sh

This is the latest test and a very simple one. Create an isofs file system with a copy of date(1). Mount the file system and run the copy of "date." This would trigger a "panic: witness_warn" as reported here: <https://people.freebsd.org/~pho/stress/log/isofs2.txt>

marcus5.sh

This is an example of a test that was a spin-off of a search for a different issue. The test triggered a problem with how VFS_SYNC() was implemented: <https://people.freebsd.org/~pho/stress/log/marcus5.txt>

md8.sh

This is a regression test for unmapped unaligned IO over a vnode-backed md(4) volume. So, with the buffer "data" page aligned, the tests are:

```
read(fd, data + 512, MAXPHYS)
write(fd, data + 512, MAXPHYS)
```

The Details

Fetch stress2 by:

```
svnLite checkout svn://svn.freebsd.org/base/user/pho/stress2
cd stress2
make
```

This will build some basic test programs in the sub-directory "testcases." All new development is done in the "misc" directory where there are currently some 400 test scenarios. These tests are often referred to as regression tests. The real value is the way they stress different corners of the kernel. Tests in the "misc" directory can either be run separately or by control of the "all.sh" script. For example, run all the tmpfs(5) scenarios once by:

```
$ ./all.sh -o tmpfs*
20150907 22:02:12 all: tmpfs2.sh
20150907 22:06:11 all: tmpfs9.sh
:
```

In Conclusion

Stress2 is a developer tool for finding problems in the FreeBSD kernel. It is not a substitute for real world testing, but just a tool for finding some of the problems.

This work is sponsored by EMC / Isilon Storage Division.

PETER HOLM (pho@FreeBSD.org) has been finding errors in FreeBSD since 1999. The Stress2 test suite is in constant development, as new tests are added as a result of bug reports or patch tests.

