

Improving kernel test coverage using stress-ng

Colin Ian King

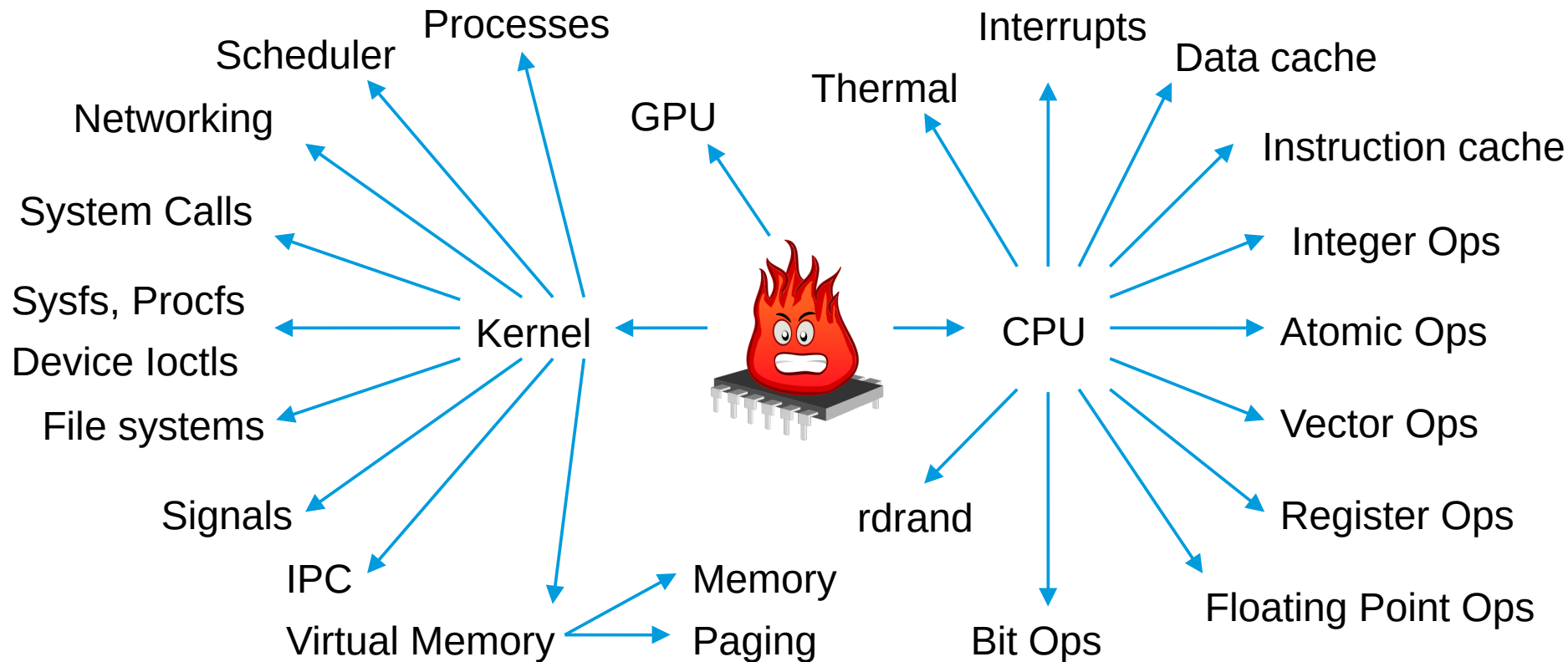
`colin.i.king@gmail.com`

Linux Plumbers Conference

Prague October 5-7 2026



What is stress-ng?



Stress-ng Features

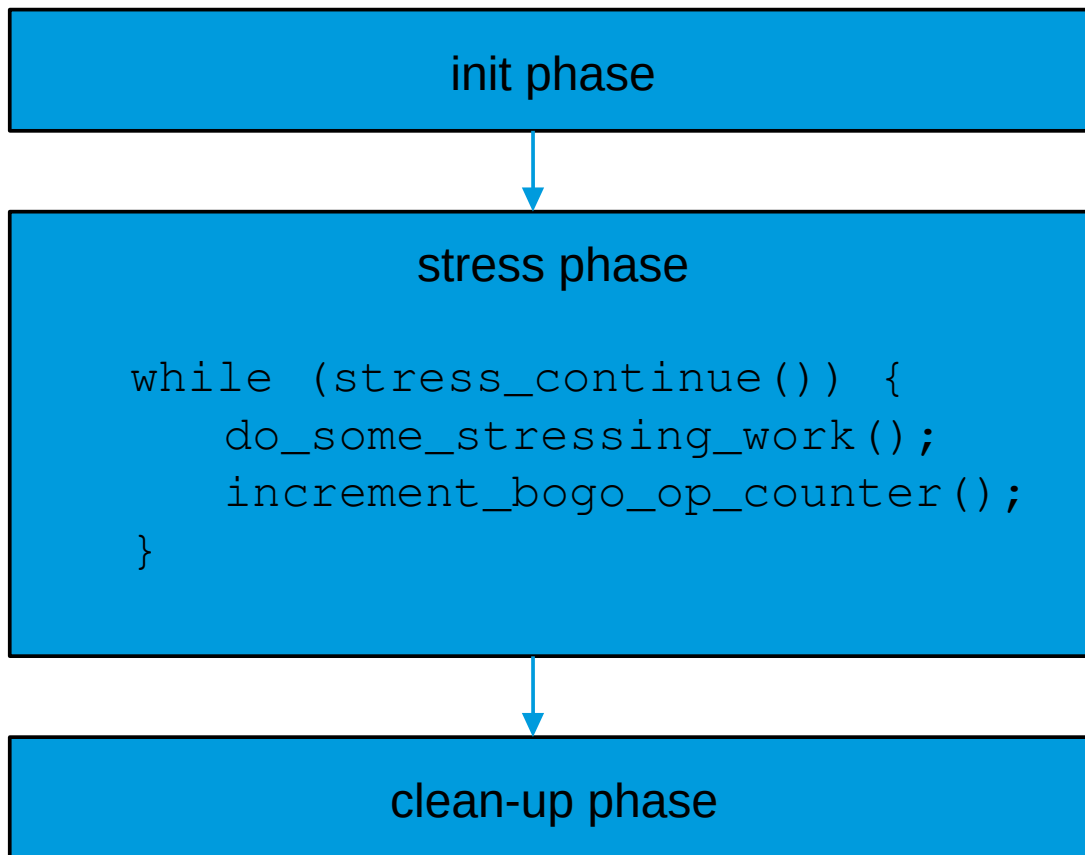
- Over 390 stress tests (aka ‘stressors’)
- “Freedom from Interference” (FFI) testing
- Portable (many architectures/operating systems, toolchains) + build-time feature detection
- Wide + deep kernel feature exercising (races, locking, resources, kernel interfaces, thermal...)
- Found over 140 OS issues (Linux/BSD/Solaris/Cygwin etc.)
- **It is not a stress-testing silver bullet!**

What is a stressor?

Normally a single process forked from stress-ng

A stressor may be one or more child processes (or one or more pthreads)

A stressor terminates on a SIGALARM timeout or has reached a maximum bogo-op end value



Using stress-ng (examples)

- `stress-ng --mmap 8 --timeout 5m`
 - Run 8 mmap stressors for 5 minutes
- `stress-ng --mmap -1 --pageswap 1 -t 1h`
 - Run mmap stressors on all online CPUs and 1 instance of swapping pages for 1 hour
- `stress-ng --class scheduler --seq 4 -t 30s`
 - Run sequentially through all scheduler stressors, 4 instances of each stressor, 30 seconds per stressor

Ongoing stress-ng development

- Add stress tests for new system calls
- Add stress tests for new kernel/user API interfaces such as new `ioctl()` and system call options/flags
- Add more exercising for new `/proc` and `/sys` files
- **Kernel gcov analysis, fill in untested core kernel code paths**
- Make regular releases (4-6 week release cadence)
- **Big thank you** for Linux Foundation ELISA project for sponsoring stress-ng work!

Kernel Coverage (gcov)

- Gcc coverage tool gcov provides kernel run-time profiling support
- Build gcov enabled kernel
- Run stress-ng tests in a QEMU virtual machine using gcov kernel
 - `stress-ng --seq 0 -t 10`
- Extract coverage data (via lcov) and check for untested code paths (genhtml and html2text)
- Figure out how to exercise untested code paths (tricky!)
- Add new tests to stress-ng

Kernel gcov Config

- `CONFIG_DEBUG_FS=y`
- `CONFIG_GCOV_KERNEL=y`
- `CONFIG_GCOV_PROFILE_ALL=y`

..and maybe add module specific configs too, e.g:

- `CONFIG_GCOV_PROFILE_RDS`

Reference:

<https://www.kernel.org/doc/html/v7.2/dev-tools/gcov.html>

genhtml output (kernel ipc)

LCOV - code coverage report

Current view: [top level](#) - ipc

Test: [kernel.info](#)

Test Date: [2026-06-02 09:40:30](#)

Coverage

Total

Hit

Lines:

75.7 %

4151

3143

Functions:

69.2 %

325

225

File	Line Coverage ↕				Function Coverage ↕		
	Rate	Total	Hit		Rate	Total	Hit
compat.c		0.0 %	32		0.0 %	4	
ipc_sysctl.c		84.1 %	107	90	81.8 %	11	9
mq_sysctl.c		90.6 %	53	48	100.0 %	6	6
mqueue.c		83.1 %	816	678	64.4 %	73	47
msg.c		72.6 %	634	460	60.0 %	50	30
msgutil.c		89.7 %	87	78	83.3 %	6	5
namespace.c		67.7 %	127	86	90.9 %	11	10
sem.c		77.3 %	1064	823	69.4 %	62	43
shm.c		69.5 %	862	599	68.7 %	67	46
syscall.c		0.0 %	43		0.0 %	3	
util.c		86.3 %	292	252	90.3 %	31	28
util.h		85.3 %	34	29	100.0 %	1	1

Untested code path (#1)

- ipc/shm.c, shmctl_down(), IPC_SET case not exercised
- Just need to add code to exercise this path, commit b3ceff08c8b1

```
1016      1141355 :      switch (cmd) {
1017      1141355 :      case IPC_RMID:
1018      1141355 :          ipc_lock_object(&shp->shm_perm);
1019      :          /* do_shm_rmid unlocks the ipc object and rcu */
1020      1141355 :          do_shm_rmid(ns, ipcp);
1021      1141355 :          goto out_up;
1022      0 :      case IPC_SET:
1023      0 :          ipc_lock_object(&shp->shm_perm);
1024      0 :          err = ipc_update_perm(&shm64->shm_perm, ipcp);
1025      0 :          if (err)
1026      0 :              goto out_unlock0;
1027      0 :          shp->shm_ctim = ktime_get_real_seconds();
1028      0 :          break;
1029      0 :      default:
1030      0 :          err = -EINVAL;
1031      0 :          goto out_unlock1;
1032      :      }
```

Untested code path (#2)

- fs/read_write.c, compat sendfile not exercised, CONFIG_COMPAT not defined. Many instances like this.

```
1435         : #ifdef CONFIG_COMPAT
1436         0 : COMPAT_SYSCALL_DEFINE4(sendfile, int, out_fd, int, in_fd,
1437         :         compat_off_t __user *, offset, compat_size_t, count)
1438         : {
1439         0 :         loff_t pos;
1440         0 :         off_t off;
1441         0 :         ssize_t ret;
1442         :
1443         0 :         if (offset) {
1444         0 :             if (unlikely(get_user(off, offset)))
1445         :                 return -EFAULT;
1446         0 :             pos = off;
1447         0 :             ret = do_sendfile(out_fd, in_fd, &pos, count, MAX_NON_LFS);
1448         0 :             if (unlikely(put_user(pos, offset)))
1449         :                 return -EFAULT;
1450         :             return ret;
1451         :         }
1452         :
1453         0 :         return do_sendfile(out_fd, in_fd, NULL, count, 0);
1454         : }
```

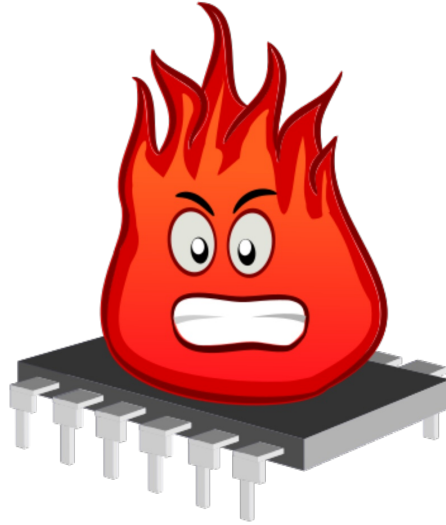
Untested code path (#3)

- ipc/shm.c, do_shmat()
- Need to force kzalloc_obj() to fail to exercise failure path..

```
1620      1493734 :      err = -ENOMEM;
1621      1493734 :      sfd = kzalloc_obj(*sfd);
1622      1490630 :      if (!sfd) {
1623          0 :          fput(base);
1624          0 :          goto out_nattch;
1625          :      }
```

How can we improve test coverage?

?



Try Kernel fault injection

- Set appropriate fault injection settings in `/sys/kernel/debug/fail*`
- Enable `--make-it-fail` option and run as root:
 - `sudo stress-ng --seq 0 -t 10 --make-it-fail`
 - This writes “1” to `/proc/self/make-it-fail` to enable per stressor process fault injection
- Reference:
<https://docs.kernel.org/fault-injection/fault-injection.html>

Preliminary Results

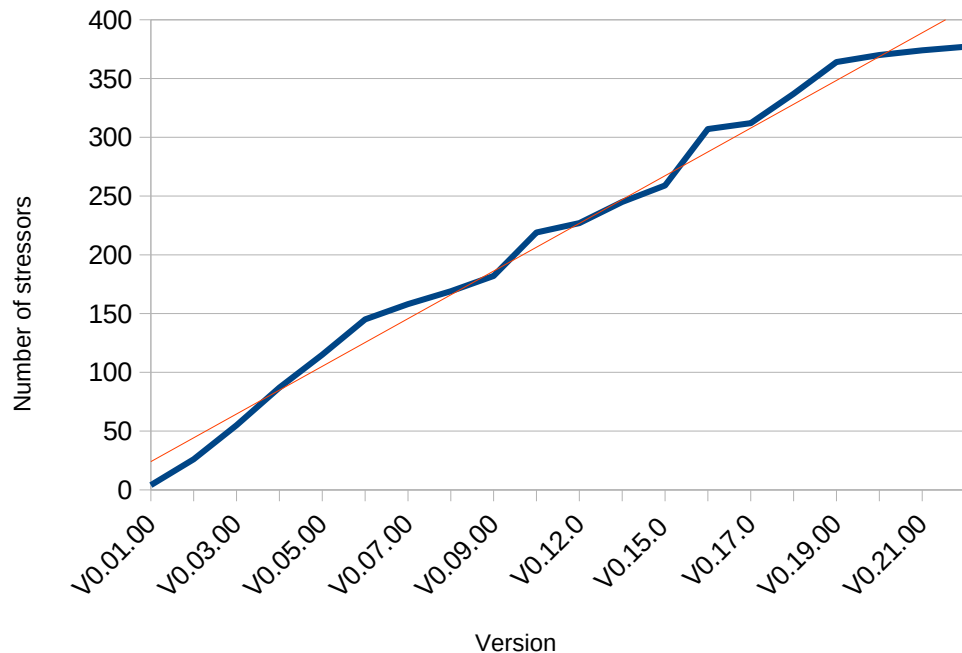
Linux 7.3-rc4			% Coverage Improvement	
	Lines	Functions	Lines	Functions
baseline	271515	24060		
<u>failslab</u>	278843	24566	2.70	2.10
<u>fail_page_alloc</u>	280877	24788	3.45	3.03
<u>fail_usercopy</u>	278452	24547	2.55	2.02
<u>fail_futex</u>	281020	24789	3.50	3.03
<u>fail_make_request</u>	276500	24435	1.84	1.56
<u>fail_function</u>	279610	24642	2.98	2.42
<u>fail_skb_realloc</u>	279615	24680	2.98	2.58

Fault Injection Summary

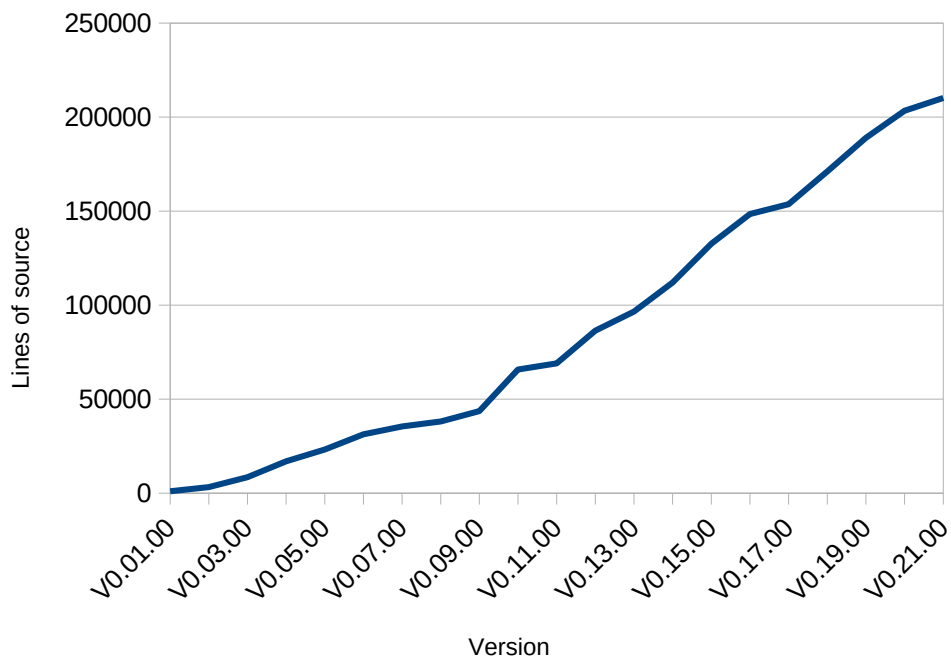
- Increases kernel coverage by ~2 to 3.5%
- Can result in testing crashes/lock-ups
 - A few are kernel related – needs identifying and fixing!
 - Some are in stress-ng (poor corner case error handling?) – needs fixing!
- Currently using SIGKILL as a workaround on stuck stressors :-)

More stressors = more coverage?

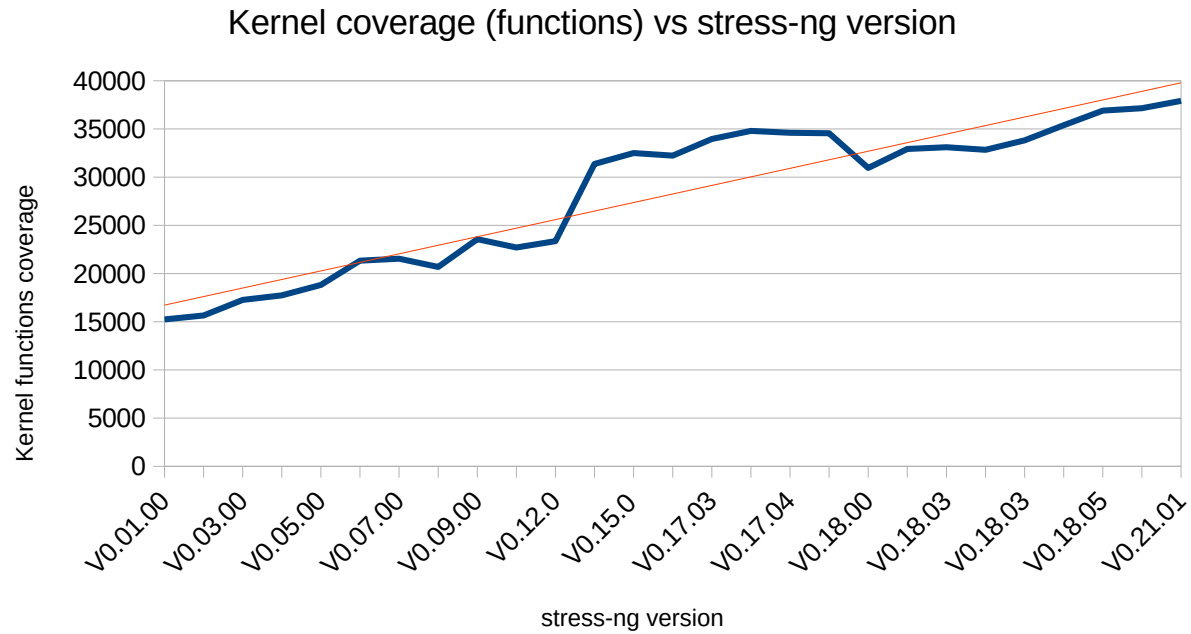
stress-ng - number of stressors vs version



stress-ng -lines of source vs version



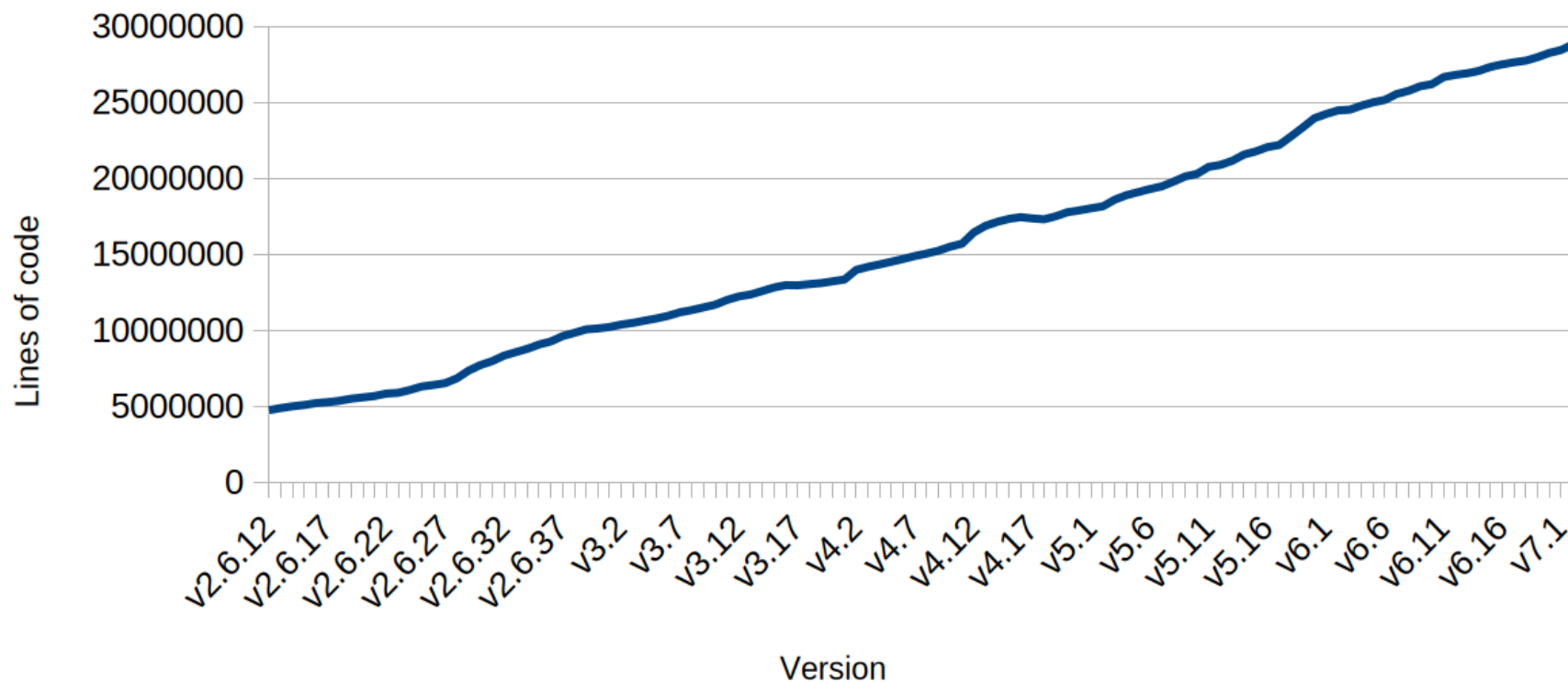
Kernel coverage trend



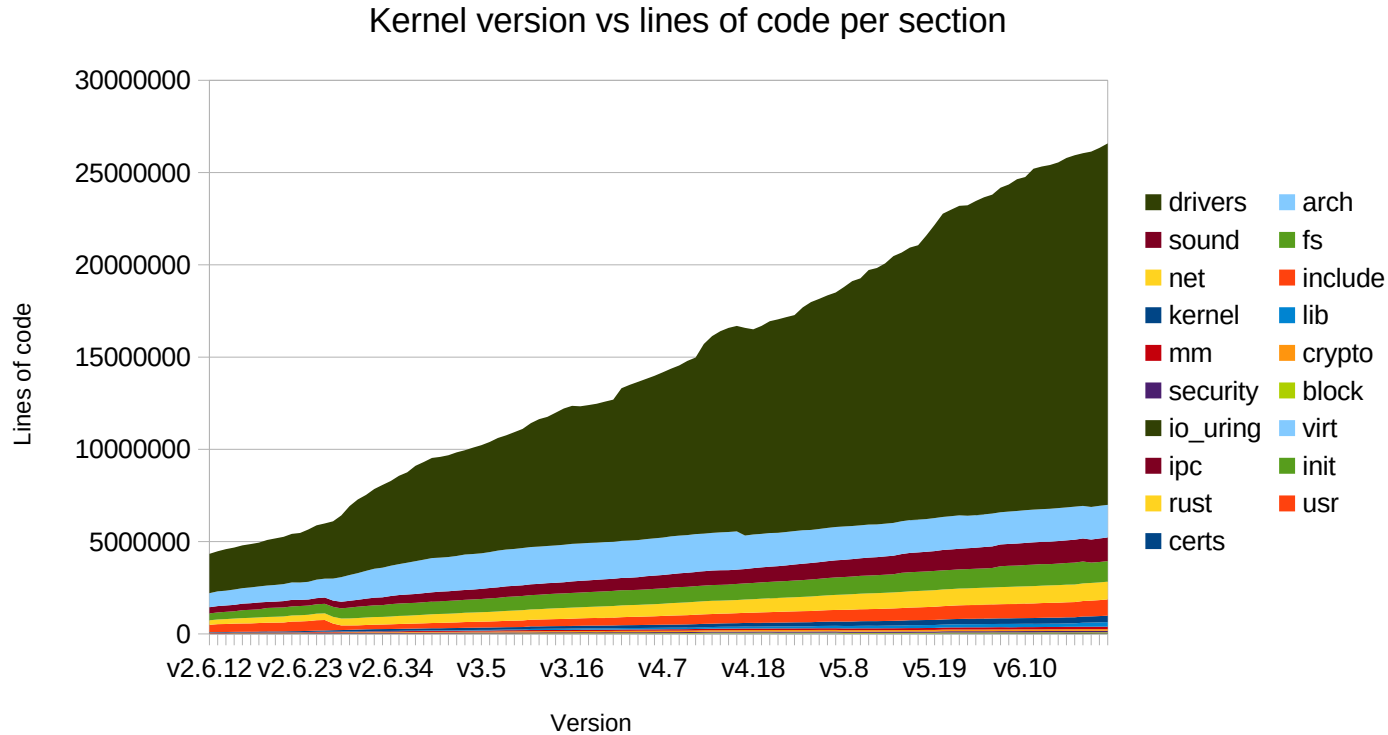
Unstoppable kernel growth!

Linux Kernel: Lines of code vs Version

(measured using sloccount)



Where is the kernel code growth?



Coverage for Kernel Drivers

- Hard to fully test drivers without hardware or suitable emulation
- Drivers normally require specialised stress-tests (/dev ioctl() interfaces, etc..)
- **Many drivers** (11,100+ .ko modules in linux 7.3-rc3, x86-64)
- Lots of functionally, poor documentation, makes exhaustive testing hard
- Consequently **poor** kernel test coverage :-(
- Needs far more attention and that takes effort (and I'm retired!)
- **Maybe not a priority?**

How well does stress-ng do?

- Linux Test Project (LTP) vs stress-ng run coverage comparison

	Lines	Functions
LTP:	575285	49127
stress-ng	499067	41130

- LTP contains test kernel modules and also runs kernel test modules
- Note: stress-ng is NOT a replacement for LTP!**
- stress-ng is designed to shake out corner case bugs when systems are under load.

The future?

- Fill in coverage test holes:
 - Holes found using coverage analysis
 - LTP vs stress-ng – what's missing in stress-ng?
 - Test driver core code (rather than specific drivers)
 - Known stress-ng coverage shortcomings:
 - sound, fuse, networking, GPU, NPU, security, cgroups

Stress-ng and ELISA

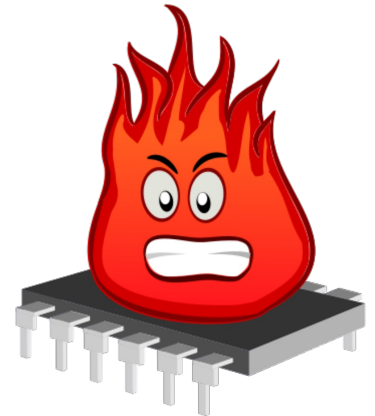
- Stress-ng now part of the Enabling Linux in Safety Applications (ELISA) project (tools)
- Thanks to ELISA for sponsoring stress-ng :-)
 - More sponsors always welcome!
- New repository location:
 - <https://github.com/stress-ng/stress-ng>
- Co-maintainer help would be awesome (aka volunteers)
- New stressors contributions welcome (patches are appreciated!)



Discussion?

Colin Ian King

- `colin.i.king@gmail.com`
- `https://github.com/stress-ng/stress-ng`
- `https://elisa.tech`



Extra: What's New?

- Specify stressors by system calls being exercised:
 - `stress-ng --exercise-syscall mmap,msync --seq 0`
- List key syscalls being exercised in stressors:
 - `Stress-ng --exercise-syscall \?`
- Specify stressors by test features (~100 features, see man page)
 - `stress-ng -exercise-feature lock-contention,chaotic-load`