Introduction to Data Science - Kernel I

©Nicholas Mc Guire <safety@osadl.org>

March 3, 2020

Linux kernel

- Started in 1991 by Linus Torvalds
- UNIX clone from scratch for i368
- Current:
  - license: GPL V2
  - version: 5.6-rc4
  - size: roughly 20 MLoC
  - developers: 20k Developers
  - change-rate: 5-6 commits per hour, 24/7, 365 days

big, complex, fast changing, a perfect target for data science to extract information from a vast and rapidly changing data set - you only can do it if you automate the hell out of it !
This is my goal for this term - lets do it. lets analyze the Linux kernel development quality !

Assessing code

We want to find out which part of this highly complex project is of high quality - with other words - can we trust Linux ? And if we can trust Linux - how much can we trust it ?
There are many questions we can ask:

- How mature is the code ?
- How well structured is it ?
- How well written is it ?
- What is the complexity of the code ?
- How widely is it used ?

All of these question tell us something, at the qualitative level, about the code - **But:** how can we quantify this ?

Assessing against existing rules

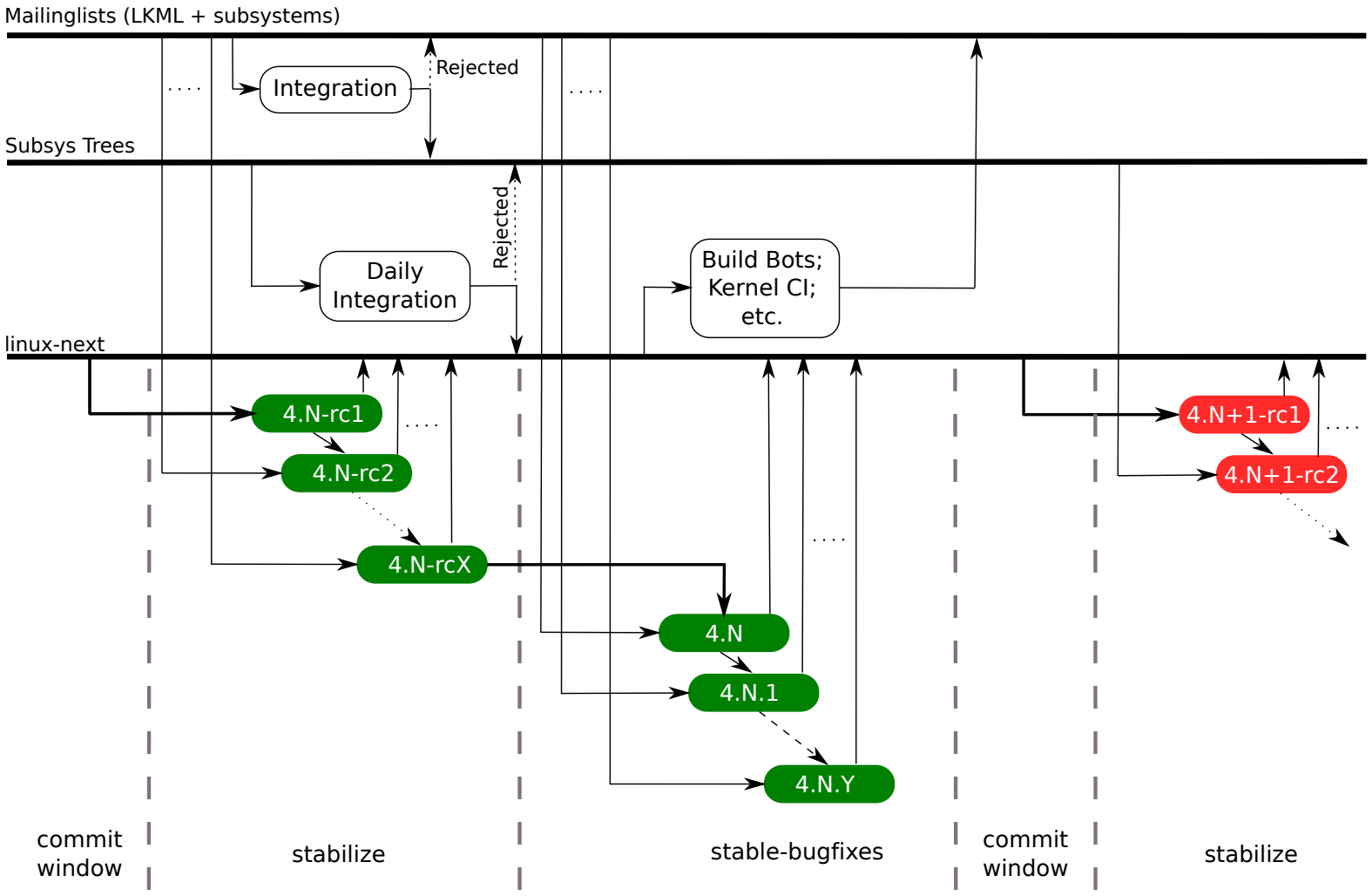
Linux kernel Procedures

- CodingStyle - simple and relatively short (40+ rules)
- checkpatch.pl - exhaustive and fussy (400+ rules)
- Amendment by tooling (sparse/coccinelle/checkpatch –strict) to cover some aspects that are not sufficiently addressable by coding style
- Amendment by procedures (SubmittingPatches,SubmitChecklist)
- Patch review procedure
- Multi-layer integration process
- Systematic compile/boot testing (build-bots/kernelCI)

First we need to look at the procedures in place then we can star thinking of how we can measure "how good is the kernel".

Linux kernel DLC

Development Life-Cycle



©Nicholas Mc Guire <safety@osadl.

Defining Attributes of interest

In data science we rarely can directly "read" the attribute of interest - we must use proxy data/proxy attributes to gain access to the properties we really want to see.
"How mature is the code"

- code that "survived" for a long time might be "better"
- code that was written by a team of people might be reviewed
- code understandability leads to good code - when is code understandable ?
  - short functions
  - only a few parameters
  - low complexity (...now go and define complexity in a measurable form!)
  - available and understandable comments
  - "good" coding style

So now we have some attributes that are closer to something we can measure - but we are **not** directly looking at code

©Nicholas Mc Guire <safety@osadl.

Example 1: following rules ?

Sha1 noted in the fixes tags should be 12 digits...
The distribution of fixes tags hash length for v4.4...v4.4.13 for all those who love statistical evidence 17.6% non-conformance ...bad ?

```
count  hash-len
  7    xxxxxxx
 11    xxxxxxxx
  8    xxxxxxxxx
 14    xxxxxxxxxx
  6    xxxxxxxxxxx
484    xxxxxxxxxxxx    <--- 12 the "proper" value
 31    xxxxxxxxxxxxx
  4    xxxxxxxxxxxxxx
  4    xxxxxxxxxxxxxxx
  5    xxxxxxxxxxxxxxxx
  1    xxxxxxxxxxxxxxxxxxxx
 19    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

So while the 8 and the 19 can be explained (those are the default length of git short sha1 and full sha1) the rest is simply wrong (probably manual) procedures.

©Nicholas Mc Guire <safety@osadl.

Example 2: Reasonable conditions

drivers/media/dvb-frontends/dib7000m.c:926 bad conditional

```
/* P_dintl_native, P_dintlv_inv,
   P_hrch, P_code_rate, P_select_hp */
value = 0;
if (1 != 0)
        value |= (1 << 6);
if (ch->hierarchy == 1)
        value |= (1 << 4);
if (1 == 1)
        value |= 1;
switch ((ch->hierarchy == 0 || 1 == 1) ?
        ch->code_rate_HP : ch->code_rate_LP) {
```


©Nicholas Mc Guire <safety@osadl.

Example 3: ...and reasonable control flow

drivers/staging/rtl8723au/hal/rtl8723a_bt-coexist.c:7264 else duplicates if

```
...
} else if (maxInterval == 2) {
        btdm_2AntPsTdma(padapter, true, 15);
        pBtdm8723->psTdmaDuAdjType = 15;
} else if (maxInterval == 3) {
        btdm_2AntPsTdma(padapter, true, 15);
        pBtdm8723->psTdmaDuAdjType = 15;
} else {
        btdm_2AntPsTdma(padapter, true, 15);
        pBtdm8723->psTdmaDuAdjType = 15;
}
```

Example 4: ...no conditions with side-effects

©Nicholas Mc Guire <safety@osadl.

drivers/ide/cmd640.c:680 redundant logic expression with side-effect

```
if (inb(0xCF8) == 0x00 && inb(0xCF8) == 0x00) {
     spin_unlock_irqrestore(&cmd640_lock, flags);
     return 1;
}
```

This has been in here since kernel 2.3.X (pre-dates git) The earlier 2.2.X kernels do not have this construct
How did this get into the kernel ?

Example 5: ..and reasonable number of parameters

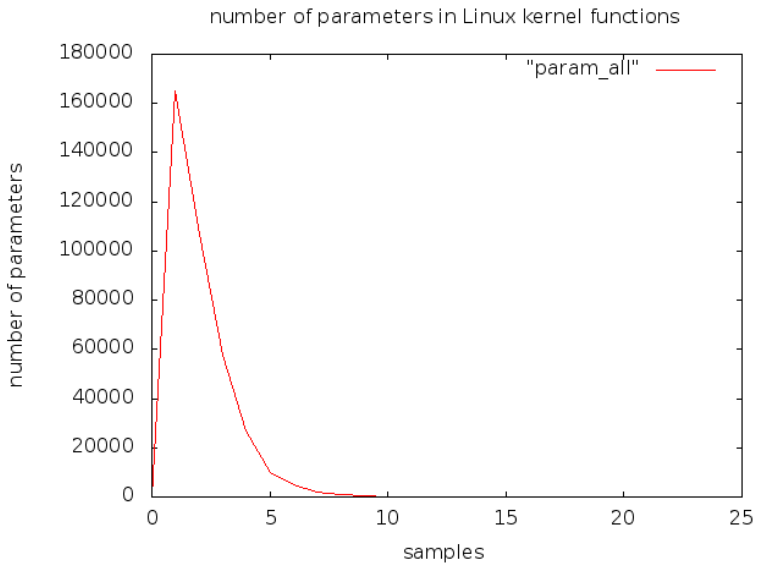
fs/ceph/caps.c:send_cap_msg,line 968 out of control parameter list

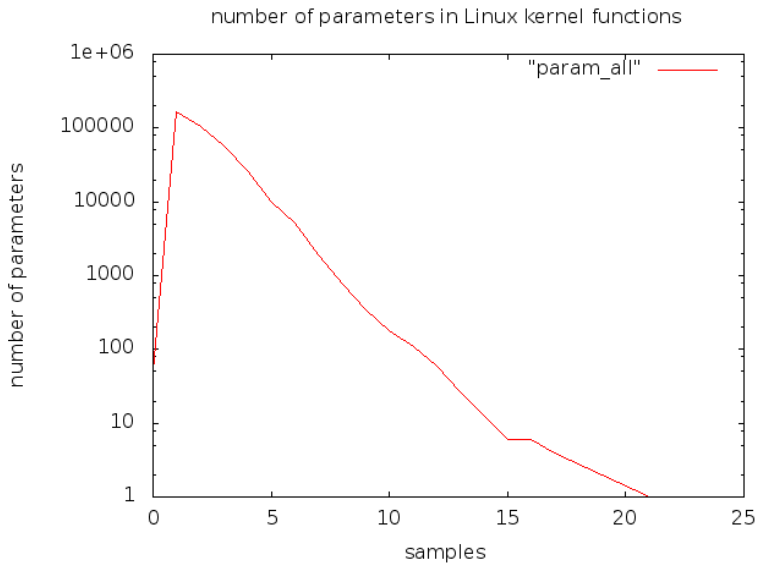
```
static int send_cap_msg(struct ceph_mds_session *session,
              u64 ino, u64 cid, int op,
              int caps, int wanted, int dirty,
              u32 seq, u64 flush_tid, u32 issue_seq, u32 mseq,
              u64 size, u64 max_size,
              struct timespec *mtime, struct timespec *atime,
              u64 time_warp_seq,
              kuid_t uid, kgid_t gid, umode_t mode,
              u64 xattr_version,
              struct ceph_buffer *xattrs_buf,
              u64 follows, bool inline_data)
{
```

Plain ugly - no excuse for this one - simply exclude ceph from the list of suitable fs.

Linux total parameter distribution

©Nicholas Mc Guire <safety@osadl.





There is a few hundred functions that are over the reasonable limit of 7-8 parameters.

The problem - your problem

- We need to build a good interface into the data first
- So we need some **git to meta-data** tools and methods
- What data we really want to see depends on the:
  - Understanding of the problem
  - Proxy variables we can construct
  - Testable models we can define
  - Level of automation we can achieve
  - Reliability of the techniques we use

Data science covers everything we need to do this - except for the domain knowledge - THAT will be your fist challenge!

Homework

1. Start building context
   - Cloning the kernel: git clone git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
   - Start reviewing linux-stable/Documentation/process/
2. Start playing with git
   - How to call simple git commands from python
   - How to collect the answers from the git command

Work together - in small teams - even if you are now fully distributed - you can do it - you can meet on IRC and work together in a git repo.