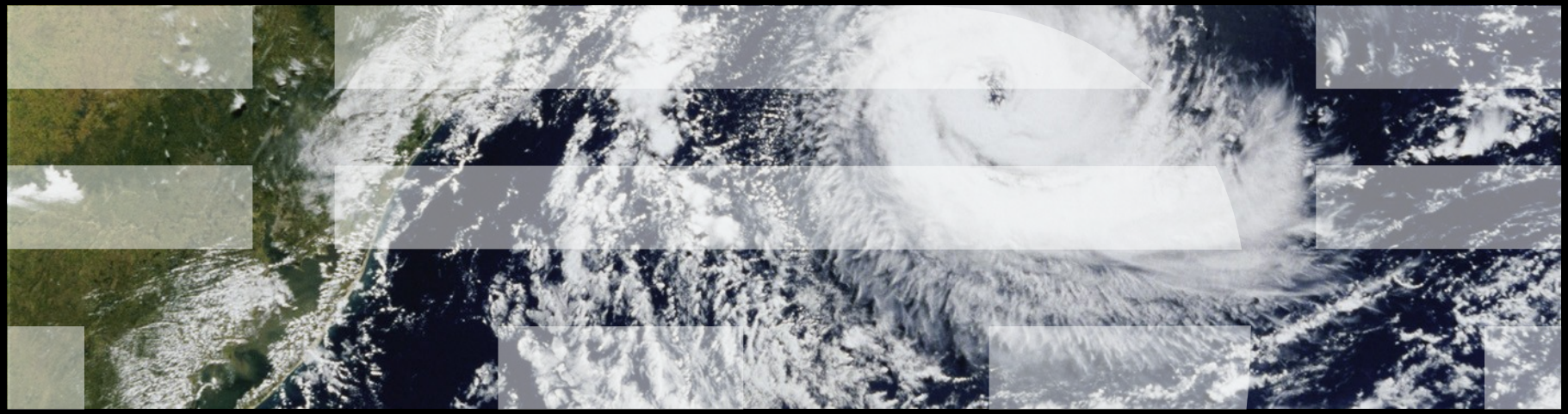
Paul E. McKenney, IBM Distinguished Engineer, Linux Technology Center
Member, IBM Academy of Technology
linux.conf.au, Geelong, Australia, February 1, 2016

Mutation Testing and RCU

Joint work with Iftekhar Ahmed, Alex Groce, and Carlos Jensen of Oregon State University



Motivation

- More than a billion instances of the Linux kernel running
- Yay, we have won!!! But...
- A million-year race-condition bug is happening three times a day across the installed base
- We need to up our validation game...
- Part of this new game might be mutation testing

What is Mutation Testing?

- Mutate the code to see if testing catches it
  - Delete lines
  - Change constants
  - Change operators
  - Negate conditionals
- Generate a separate source file for each mutation
  - Thousands of mutants for RCU
  - Long test time, also need manual analysis of the cases that don't fail
    - Should the test have caught it, and if so, how to fix the test?

Replacing a Constant

- In data declaration:
  - /* MUTANT (rep_const) */ .level = { &sname##_state.node[1] },
  - Was zero... Test catches this one.
- Some have no effect:
  - /* MUTANT (rep_const) */ while (2) {
  - Skip testing things that generate identical binary to original
- Some have no effect in more complicated ways:
  - /* MUTANT (rep_const) */ return 2;
  - Manual inspection required, perhaps use compiler tricks later
- Some have no effect in *really* complicated ways:
  - /* MUTANT (rep_const) */ .gpnum = 0UL - -1UL,

Advantages and Disadvantages of Mutation Testing

- No specification required
  - Which is a good thing, because an RCU specification would be as big and as buggy as the Linux-kernel RCU implementation
- Another way to put this: Mutation testing leverages the specification implicit in the compiler, the assertions, and rcutorture
- Some mutants introduce low-probability race condition
  - Extreme runtimes for those cases
- Some mutants require significant manual analysis
  - That would be me...

Mutation Testing Strategy

- Discard uninteresting mutants as quickly as possible
- Generated 3393 mutants for Tiny RCU
  - Resulted in 3394 kernel compilations
- Narrowed search field:
  - 378 failed to compile
  - 321 generated same binary as original
  - 746 generated same binary as some other mutant
- 2339 mutants remaining (69% of total)
  - 2180 failed running rcutorture on two-CPU x86 for two minutes
- 159 required manual analysis (4.68% or total)
  - That wouild be me...

Result of Manual Analysis

- Patch 1: Update rcutorture to cover SRCU cleanup path
- Patch 2: Make rcutorture test Tiny RCU-bh
- Patch 3: Fix Tiny RCU callback list bug
  - Bug detected by running with Patch 2

- So mutation testing has had some good results
- Next step: Try it on Tree RCU

Preliminary Results From Tree RCU Mutants

- Seven potential rcutorture improvements:
  - Ensure callback lists are non-empty when CPU goes offline
    - (Though I suspect that callback flooding does this.)
  - Enhance rcutorture to detect performance and realtime degradations
    - (But rcutorture intentionally does many degrading things.)
  - Review rcutorture's RCU priority boost testing
  - Invoke call_rcu() from an idle CPU when torture testing
  - Sometimes invoke call_rcu() with interrupts disabled
  - Come up with some way to test counter-wrap conditions
    - (Could be tough... Shrink counters to eight bits?)
  - Add scenarios that adjust rcu_fanout_leaf to cover more boot code
- Stats and overview TBD

What is Next?

- Actually do some of the rcutorture improvements
- Heavier-duty torture testing of mutants
- Introduce parallel testing to speed up the process
  - Running 16 rcutorture scenarios a few thousand times each does not happen overnight...
- Improved automated detection of uninteresting mutants
- Add additional classes of mutants

Legal Statement

- This work represents the view of the author and does not necessarily represent the view of IBM.
- IBM and IBM (logo) are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.

Questions?