Detecting RCU Bugs

Dhaval Giani
(With a lot of input from Paul McKenney and Frederic Weisbecker)

Not everyone is Paul McKenney*!

Not everyone is Paul McKenney*!

* or Steven Rostedt or Frederic Weisbecker or Mathieu Desnoyers or someone else (in this room or otherwise) who understand RCU well.

Not everyone is Paul McKenney*!

IOW it is easy to write RCU bugs

* or Steven Rostedt or Frederic Weisbecker or Mathieu Desnoyers or someone else (in this room or otherwise) who understand RCU well.

Defining RCU

RCU guarantees existence of consistent data within an RCU read critical section

Defining RCU: Correctness

Defining RCU: Correctness

- RCU protected data is consistent

Defining RCU: Correctness

- RCU protected data is consistent

- RCU protected data is read only

Defining RCU: Correctness

- RCU protected data is consistent
- RCU protected data is read only
- Garbage collection must take place

Defining RCU: Consistency

Defining RCU: Consistency

A data structure is defined to be consistent if (maybe iff) it meets a set of programmer defined invariants for that structure

Defining RCU: Correctness

- RCU protected data is consistent
- RCU protected data is read only
- Garbage collection must take place

Defining RCU: Correctness

- ~~RCU protected data is consistent~~

- RCU protected data is read only

- Garbage collection must take place

Defining RCU: Correctness

- ~~RCU protected data is consistent~~
- RCU protected data is guaranteed to remain in existence

- RCU protected data is read only

- Garbage collection must take place

Defining RCU: Correctness

- ~~RCU protected data is consistent~~
- RCU protected data is guaranteed to remain in existence

- ~~RCU protected data is read only~~

- Garbage collection must take place

Defining RCU: Correctness

- ~~RCU protected data is consistent~~
- RCU protected data is guaranteed to remain in existence

- ~~RCU protected data is read only~~
- RCU protected data is read only or a well defined update policy

- Garbage collection must take place

RCU Bugs

- Correctness Bugs
  - The ones we care about

- Performance Bugs
  - Blame Paul for these!

Correctness bugs

- Garbage collection must take place after grace period
  - Run too early
  - Access protected data outside CS
  - CS never ends
- RCU protected data is read only
  - Protected data is modified
- RCU protected data is consistent
  - Data is inconsistent

Detecting Pointer Leaks

Pre-existing techniques for race detection

Lockset?

No locks

What if we convert RCU into locks?

RCU <-> Locks

- Easy to convert rwlocks to RCU
- What about converting RCU to rwlocks?
- Need a new transformation
  - Need functional equivalence

RCU <-> rwlocks

RCU <-> rwlocks

- Multiple versions of given data item
  - Each version is protected by its own rwlock

RCU <-> rwlocks

- Multiple versions of given data item
  - Each version is protected by its own rwlock
- rcu_read_lock()
  - Let the system know a CS has started

RCU <-> rwlocks

- Multiple versions of given data item
  - Each version is protected by its own rwlock
- rcu_read_lock()
  - Let the system know a CS has started
- rcu_dereference
  - Grab the private rwlock in read mode

RCU <-> rwlocks

- Multiple versions of given data item
  - Each version is protected by its own rwlock
- rcu_read_lock()
  - Let the system know a CS has started
- rcu_dereference
  - Grab the private rwlock in read mode
- rcu_read_unlock()
  - Drop all the private rwlocks and end the CS

RCU <-> rwlocks

- Multiple versions of given data item
  - Each version is protected by its own rwlock
- rcu_read_lock()
  - Let the system know a CS has started
- rcu_dereference
  - Grab the private rwlock in read mode
- rcu_read_unlock()
  - Drop all the private rwlocks and end the CS
- Cleanup
  - Only place where you go write mode

RCU <-> rwlocks

- Multiple versions of given data item
  - Each version is protected by its own rwlock
- rcu_read_lock()
  - Let the system know a CS has started
- rcu_dereference
  - Grab the private rwlock in read mode
- rcu_read_unlock()
  - Drop all the private rwlocks and end the CS
- Cleanup
  - Only place where you go write mode
  - Should NEVER block!

RCU <-> rwlocks

RCU <-> rwlocks

- Very Complex!

RCU <-> rwlocks

- Very Complex!

- Fails the paulmck test
  - Way too many corner cases

RCU <-> rwlocks

- Very Complex!
- Fails the paulmck test
  - Way too many corner cases
- What else could we do?

RCU <-> rwlocks

- Very Complex!
- Fails the paulmck test
  - Way too many corner cases
- What else could we do?
  - I really wish I could watch all RCU protected data!

Unlimited Watchpoints

Unlimited Watchpoints

- My apologies to other architectures

Unlimited Watchpoints

- My apologies to other architectures

- x86-64 uses only 48 bits for addressing
  - The upper 16 bits are always set
  - Else, we have a GP

Unlimited Watchpoints

- My apologies to other architectures
- x86-64 uses only 48 bits for addressing
  - The upper 16 bits are always set
  - Else, we have a GP
- What if I was expecting a GP?

RCU and Watchpoints

RCU and Watchpoints

- Poison all pointers inside rcu_assign_pointer

RCU and Watchpoints

- Poison all pointers inside rcu_assign_pointer
- Add data during rcu_dereference

RCU and Watchpoints

- Poison all pointers inside rcu_assign_pointer
- Add data during rcu_dereference
- When we enter a GPF, check!

Current Status

- Basic tracepoints are ready
- Basic algorithm to detect RCU issues designed
- Working on emulating instructions

Questions?