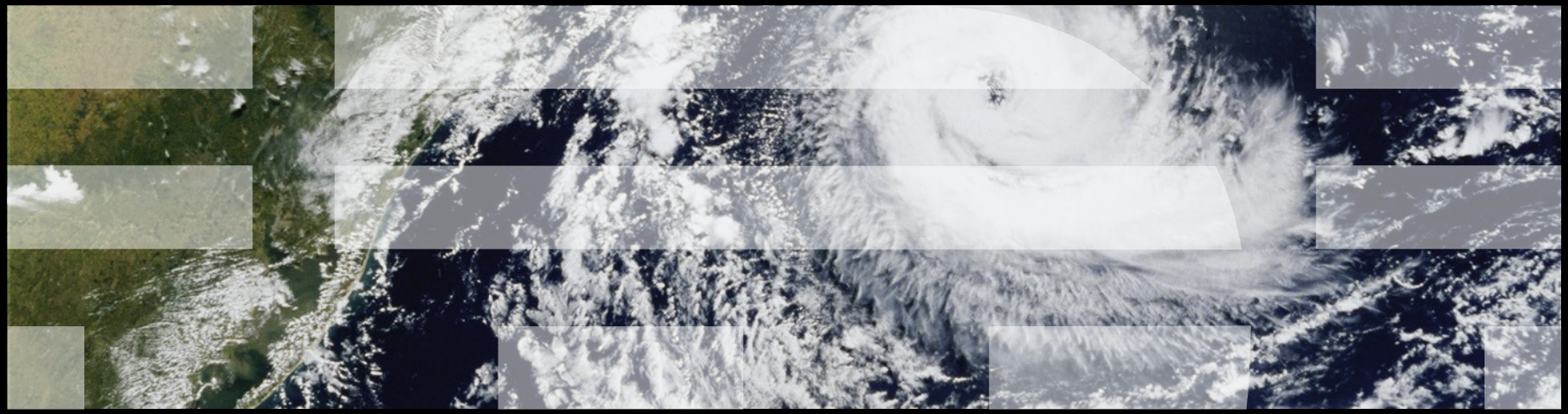
Paul E. McKenney, IBM Distinguished Engineer, Linux Technology Center
Member, IBM Academy of Technology
linux.conf.au Lightning Talk, January 26, 2018

What Happened to the Linux-Kernel Memory Model?

Joint work with Jade Alglave, Luc Maranget, Andrea Parri, and Alan Stern



Purpose: Analyze “Litmus Tests”

Thread 0:

```
WRITE_ONCE(*x0, 1);
r1 = READ_ONCE(x1);
```

Thread 1:

```
WRITE_ONCE(*x1, 1);
r1 = READ_ONCE(x2);
```

Thread 2:

```
WRITE_ONCE(*x2, 1);
r1 = READ_ONCE(x0);
```

“Exists” Clause

```
(0:r1=0 /\ 1:r1=0 /\ 1:r1=0)
```

Who Cares About Memory Models, and If So, Why???

- Hoped-for benefits of a Linux-kernel memory model
  - Memory-ordering education tool
  - Core-concurrent-code design aid
  - Ease porting to new hardware and new toolchains
  - Basis for additional concurrency code-analysis tooling
    - For example, CBMC and Nidhugg (CBMC now part of rcutorture)
- Likely drawbacks of a Linux-kernel memory model
  - Extremely limited code size
    - Analyze concurrency core of algorithm
    - Maybe someday automatically identifying this core
    - Perhaps even automatically stitch together multiple analyses (dream on!)
  - Limited types of operations (no function call, structures, call_rcu(), …)
    - Can emulate some of these
    - We expect that tools will become more capable over time
    - But I wouldn't suggest holding your breath waiting for CPU hotplug

LCA 2017 Model Capabilities

- READ_ONCE() and WRITE_ONCE()
- smp_store_release() and smp_load_acquire()
- rcu_assign_pointer()
- rcu_dereference() and lockless_dereference()
- rcu_read_lock(), rcu_read_unlock(), and synchronize_rcu()
  - Also synchronize_rcu_expedited(), but same as synchronize_rcu()
- smp_mb(), smp_rmb(), smp_wmb(), and smp_read_barrier_depends()
- xchg(), xchg_relaxed(), xchg_release(), and xchg_acquire()
- spin_trylock() and spin_unlock() prototypes in progress

Current Model Capabilities (linux-kernel.def)

- READ_ONCE() and WRITE_ONCE()
- smp_store_release() and smp_load_acquire()
- rcu_assign_pointer()
- rcu_dereference() and lockless_dereference()
- rcu_read_lock(), rcu_read_unlock(), and synchronize_rcu()
  - Also synchronize_rcu_expedited(), but same as synchronize_rcu()
- smp_mb(), smp_rmb(), smp_wmb(), smp_read_barrier_depends(), **smp_mb__before_atomic(), smp_mb__after_atomic(), and smp_mb__after_spinlock()**
- **Lots of atomic operations (see next slide)**
- spin_lock() and spin_unlock() **(but need more testing)**

Current Atomics Capabilities (linux-kernel.def)

- xchg(), xchg_relaxed(), xchg_release(), xchg_acquire(), cmpxchg(), cmpxchg_relaxed(), cmpxchg_acquire(), cmpxchg_release(), atomic_read(), atomic_set(), atomic_read_acquire(), atoimc_set_release(), atomc_add(), atomic_sub(), atomic_inc(), atomic_dec(), atomic_add_return(), atomic_add_return_relaxed(), atomic_add_return_acquire(), atomic_add_return_release(), atomic_fetch_add(), atomic_fetch_add_relaxed(), atomic_fetch_add_acquire(), atomic_fetch_add_release(), atomic_inc_return(), atomic_inc_return_relaxed(), atomic_inc_return_acquire(), atomic_fetch_inc_release(), atomic_sub_return(), atomic_sub_return_relaxed(), atomic_sub_return_acquire(), atomic_sub_return_release(), atomic_fetch_sub(), atomic_fetch_sub_relaxed(), atomic_fetch_sub_acquire(), atomic_fetch_sub_release(), atomic_dec_return(), atomic_dec_return_relaxed(), atomic_dec_return_acquire(), atomic_dec_return_release(), atomic_fetch_dec(), atomic_fetch_dec_relaxed(), atomic_fetch_dec_acquire(), atomic_fetch_dec_release(), atomic_sub_and_test(), atomic_dec_and_test), atomic_inc_and_test(), atomic_add_negative()

… And Limitations

- But there are some limitations:
  - Compiler optimizations not modeled
  - ~~No arithmetic~~
  - Single access size, no partially overlapping accesses
  - No arrays or structs (but can do trivial linked lists)
  - No dynamic memory allocation
  - Read-modify-write atomics: ~~Only xchg() and friends for now~~
  - ~~No locking (but can emulate locking operations with xchg())~~
  - No interrupts, exceptions, I/O, or self-modifying code
  - No functions
  - No asynchronous RCU grace periods, but can emulate them:
    - Separate thread with release-acquire, grace period, and then callback code
- Something about wanting the model to execute in finite time...

How to Run Models

- Download herd tool as part of diy toolset
  - http://diy.inria.fr/sources/index.html
- Build as described in INSTALL.txt
  - Need ocaml v4.02.0 or better: http://caml.inria.fr/download.en.html
    - Or install from your distro (easier and faster!)
- Run various litmus tests:
  - herd7 -conf linux-kernel.cfg litmus-tests/R+mbonceonces.litmus
  - herd7 -conf linux-kernel.cfg litmus-tests/MP+polocks.litmus
- Other required files:
  - linux.def: Support pseudo-C code
  - linux-kernel.cfg: Specify LKMM
  - linux-kernel.bell: “Bell” file defining events and relationships
  - linux-kernel.cat: “Cat” file defining actual memory model
  - linux-kernel.def: C-to-LISA mappings
  - *.litmus: Litmus tests

https://github.com/aparri/memory-model.git

But When Will LKMM be in the Linux Kernel?

But When Will LKMM be in the Linux Kernel?

- Versions one and two of patch posted to LKML:
  - http://lkml.kernel.org/r/20171113184031.GA26302@linux.vnet.ibm.com
  - http://lkml.kernel.org/r/20180119035855.GA29296@linux.vnet.ibm.com

But When Will LKMM be in the Linux Kernel?

- Versions one and two of patch posted to LKML:
  - http://lkml.kernel.org/r/20171113184031.GA26302@linux.vnet.ibm.com
  - http://lkml.kernel.org/r/20180119035855.GA29296@linux.vnet.ibm.com
- Some interest from a few relevant maintainers:
  - Reviewed-by: Boqun Feng <boqun.feng@gmail.com>
  - Acked-by: Will Deacon <will.deacon@arm.com>
  - Acked-by: Peter Zijlstra <peterz@infradead.org>
  - Acked-by: Nicholas Piggin <npiggin@gmail.com>
  - Acked-by: David Howells <dhowells@redhat.com>
  - Acked-by: "Reshetova, Elena" <elena.reshetova@intel.com>
  - Acked-by: Michal Hocko <mhocko@suse.com>

But When Will LKMM be in the Linux Kernel?

- Versions one and two of patch posted to LKML:
  - http://lkml.kernel.org/r/20171113184031.GA26302@linux.vnet.ibm.com
  - http://lkml.kernel.org/r/20180119035855.GA29296@linux.vnet.ibm.com
- Some interest from a few relevant maintainers:
  - Reviewed-by: Boqun Feng <boqun.feng@gmail.com>
  - Acked-by: Will Deacon <will.deacon@arm.com>
  - Acked-by: Peter Zijlstra <peterz@infradead.org>
  - Acked-by: Nicholas Piggin <npiggin@gmail.com>
  - Acked-by: David Howells <dhowells@redhat.com>
  - Acked-by: "Reshetova, Elena" <elena.reshetova@intel.com>
  - Acked-by: Michal Hocko <mhocko@suse.com>
- Sent a pull request yesterday, so ***maybe*** v4.16
  - http://lkml.kernel.org/r/20180125093440.GA875@linux.vnet.ibm.com

But When Will LKMM be in the Linux Kernel?

- Versions one and two of patch posted to LKML:
  - http://lkml.kernel.org/r/20171113184031.GA26302@linux.vnet.ibm.com
  - http://lkml.kernel.org/r/20180119035855.GA29296@linux.vnet.ibm.com
- Some interest from a few relevant maintainers:
  - Reviewed-by: Boqun Feng <boqun.feng@gmail.com>
  - Acked-by: Will Deacon <will.deacon@arm.com>
  - Acked-by: Peter Zijlstra <peterz@infradead.org>
  - Acked-by: Nicholas Piggin <npiggin@gmail.com>
  - Acked-by: David Howells <dhowells@redhat.com>
  - Acked-by: "Reshetova, Elena" <elena.reshetova@intel.com>
  - Acked-by: Michal Hocko <mhocko@suse.com>
- Sent a pull request yesterday, so ***maybe*** v4.16
  - http://lkml.kernel.org/r/20180125093440.GA875@linux.vnet.ibm.com
- Here is hoping!!!

Legal Statement

- This work represents the view of the authors and does not necessarily represent the view of their employers.
- 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.