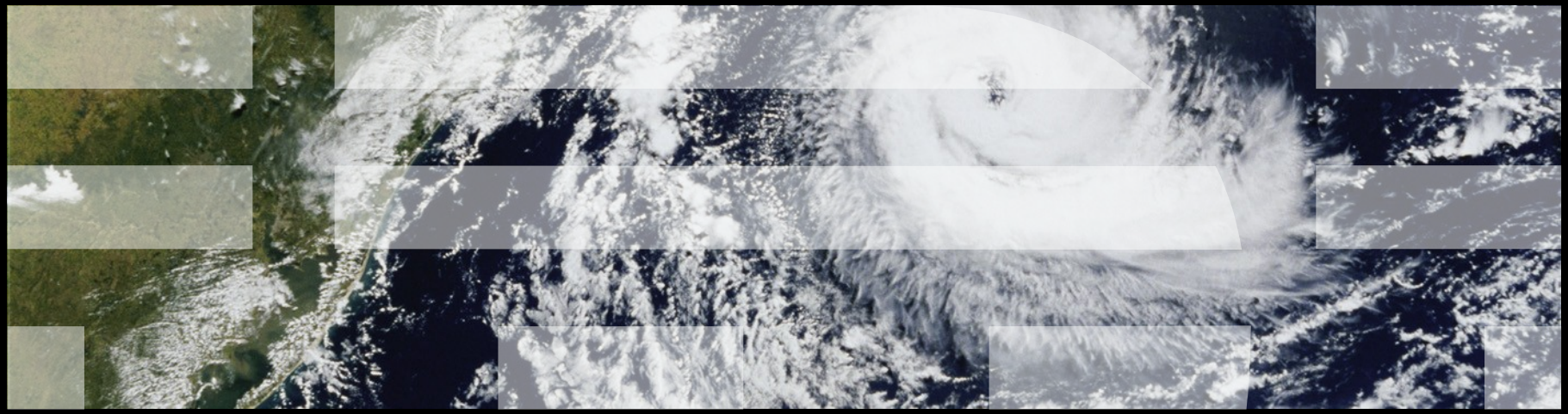
Paul E. McKenney, IBM Distinguished Engineer, Linux Technology Center
Member, IBM Academy of Technology
Beaver Barcamp, April 16, 2016

Practical Experience With Formal Verification Tools

Report from Royal Society Verified Trustworthy Software Systems Meeting
April 4-7, 2016



Overview

- Why do I care about formal verification?
- Royal Society Verified Trustworthy Software Systems meeting
  - April 4-5
- Verified Trustworthy Software Systems specialist meeting
  - April 6-7 plus concurrency discussion on April 8
- Formal verification from a Linux-kernel RCU viewpoint
- Challenges and Plan

Why Do I Care About Formal Verification?

Two Definitions and a Consequence

- A non-trivial software system contains at least one bug
- A reliable software system contains no known bugs

- Therefore, any non-trivial reliable software system contains at least one bug that you don't know about
- Yet there are more than a billion users of the Linux kernel
  - In practice, validation is about reducing risk
  - Can formal verification now take a front-row seat in this risk reduction?
- ***What would need to happen for me to include formal verification in my Linux-kernel RCU regression testing?***

Current Linux-Kernel Regression Testing

- Stress-test suite example: “rcutorture”
  - http://lwn.net/Articles/154107/, http://lwn.net/Articles/622404/
- “Intelligent fuzz testing”: “trinity”, “syzkaller”
  - http://codemonkey.org.uk/projects/trinity/
  - https://github.com/google/syzkaller/wiki/Found-Bugs
- Test suite including static analysis: “0-day test robot”
  - https://lwn.net/Articles/514278/
- Integration testing: “linux-next tree”
  - https://lwn.net/Articles/571980/
- **Above is old technology – but not entirely ineffective**
  - 2010: wait for -rc3 or -rc4. 2013: No problems with -rc1
- Formal verification in design, but not in regression testing
  - http://lwn.net/Articles/243851/, https://lwn.net/Articles/470681/, https://lwn.net/Articles/608550/

Why Formal Verification?

- At least one billion embedded Linux devices
  - A bug that occurs once per million years manifests three times per day
  - But assume a 1% duty cycle, 10% in the kernel, and 1% of that in RCU
  - 10,000 device-years of RCU per year: p(RCU) = 10^{-5}
- At ***least*** 80 million Linux servers
  - A bug that occurs once per million years manifests twice per month
  - Assume 50% duty cycle, 10% in the kernel, and 1% of that in RCU
  - 40,000 system-years of RCU per year: p(RCU) = $5(10^{-4})$
- Races between RCU event pairs, p(bug)=0.01p(RCU):
  - N-CPU probability of race: $1-(1-p(bug))^N-Np(1-p(bug))^{(N-1)}$
  - Assume rcutorture p(RCU)=1, compute rcutorture speedup:
    - Embedded: 10^{12}: 7.9 days of rcutorture testing covers one year
    - Server: $4(10^6)$: 21.9 *years* of rcutorture testing covers one year
    - Linux kernel releases are only about 60 days apart: RCU is moving target

https://www.quora.com/How-many-servers-exist-in-the-world

How Does RCU Work Without Formal Verification?

- What is validation strategy for 80M server systems?
  - Other failures mask those of RCU, including hardware failures
    - I know of no human artifact with a million-year MTBF
  - Increasing CPUs on test system increases race probability
    - And many systems have relatively few CPUs
  - Rare but critical operations can be forced to happen more frequently
    - CPU hotplug, expedited grace periods, RCU barrier operations...
  - Knowledge of possible race conditions allows targeted tests
    - Plus other dirty tricks learned in 25 years of testing concurrent software
  - Formal verification ***is*** used for some aspects of RCU design
    - Dyntick idle, sysidle, NMI interactions
- But it would be valuable to use formal verification as part of RCU's regression testing!

Royal Society
Verified Trustworthy Software Systems Meeting

But Does Formal Verification Work in Real World? View From Royal Society Meeting...

- Many other speakers focused on speed and scalability
  - "Teatime, lunchtime, or bedtime": Welcome change from past events
  - More formal-verification academics gaining industry experience
    - "Deliver value first and save the sanctimonious lectures!"
    - Others wish to drive formal-verification usage via liability legislation ...
    - … but clearly have not applied verification to their proposed legal changes!
- Extremely effective in restricted environments (safety critical)
  - Neil White, Altran UK: 0.04 bugs per KLoC
    - Still do testing, but bug found in test is considered process fault
  - Numerous avionics projects withstood heavy penetration testing
    - "This must be the most secure UAV on the planet"
- Many speakers noted risks of assumptions and specification
  - "Tell attacker exactly what to do to break software!"
  - "What attack does strong typing defend against?" (Developer errors!)

Verified Trustworthy Software Systems Specialist Meeting

But Does Formal Verification Work in Real World? View From Specialist Meeting (1/2)

- Gernot Heiser and Gerwin Klein discussed seL4
  - Adding concurrency, but with BKL to reduce verification effort(!)
  - Claimed that each lock doubles verification overhead
- John Launchbury, Director DARPA I2O
  - “Cyber retrofit” by analogy with “seismic retrofit”
  - Formal verification for new machine-learning techniques?
  - “Verified secure software cannot give up significant performance or functionality”
  - Three levels of formal verification, all are important:
    - “Mathematics”: Might one day fundamentally change the way we think about formal verification
    - “Physics”: Might affect practitioners in 3-5 years
    - “Engineering”: Immediate impact in practice
  - “Don't be naïve!!!”

But Does Formal Verification Work in Real World? View From Specialist Meeting (2/2)

- Adam Chlipala: Correct by construction approach
- Georges Gonthier: Books of proofs to electronic form
- Mark S. Miller, Arthur Charguéraud: JavaScript!!!
- Viktor Vafeiadis: Verify concurrent C/C++
- Peter Sewell: C/C++: Standards vs. compilers vs. developers
- Warren Hunt and Anna Slobodova: x86 specification
  - Includes automated proof by induction for loops!
- Daniel Kroening and Martin Brain: CBMC futures
- As for me...

Formal Verification From A Linux-kernel RCU Viewpoint

Formal Verification and Regression Testing: Requirements

(1) Either automatic translation or no translation required
- Automatic discarding of irrelevant portions of the code
- Manual translation provides opportunity for human error!

(2) Correctly handle environment, including memory model
- The QRCU validation benchmark is an excellent cautionary tale

(3) Reasonable memory and CPU overhead
- Bugs must be located in practice as well as in theory
- Linux-kernel RCU is 15KLoC (plus 5KLoC tests) and release cycles are short

(4) Map to source code line(s) containing the bug
- "Something is wrong somewhere" is not helpful: I already **know** bugs exist

(5) Modest input outside of source code under test
- Preferably glean much of the specification from the source code itself (empirical spec!)
- Specifications are software and can have their own bugs

(6) Find relevant bugs
- Low false-positive rate, weight towards likelihood of occurrence (fixes create bugs!)

Promela/spin: Design-Time Verification

- 1993: Shared-disk/network election algorithm (pre-Linux)
  - Hadn't figured out bug injection: Way too trusting!!!
  - Single-point-of failure bug in specification: Fixed during coding
    - But fix had bug that propagated to field: Cluster partition
  - **Conclusion**: Formal verification is trickier than expected!!!
- 2007: RCU idle-detection energy-efficiency logic
  - (http://lwn.net/Articles/243851/)
  - Verified, but much simpler approach found two years later
  - **Conclusion**: The need for formal verification is a symptom of a too-complex design
- 2012: Verify userspace RCU, emulating weak memory
  - Two independent models (Desnoyers and myself), **bug injection**
- 2014: NMIs can nest!!! Affects energy-efficiency logic
  - Verified, and no simpler approach apparent thus far!!!
  - Note: Excellent example of **empirical specification**

PPCMEM and Herd

- Verified suspected bug in Power Linux atomic primitives
- Found bug in Power Linus spin_unlock_wait()
- Verified ordering properties of locking primitives
- Excellent memory-ordering teaching tools
  - Starting to be used more widely within IBM as a design-time tool
- PPCMEM: (http://lwn.net/Articles/470681/)
  - Accurate but slow
- Herd: (http://lwn.net/Articles/608550/)
  - Faster, but some correctness issues with RMW atomics and lwsync
  - Work in progress: Formalize Linux-kernel memory model
    - With Alglave, Maranget, Parri, and Stern, plus lots of architects
    - Hopefully will feed into improved tooling

Alglave, Maranget, Pawan, Sarkar, Sewell, Williams, Nardelli:
"PPCMEM/ARMMEM: A Tool for Exploring the POWER and ARM Memory Models"
Alglave, Maranget, and Tautschnig: "Herding Cats: Modelling, Simulation, Testing, and Data-mining for Weak Memory"

C Bounded Model Checker (CBMC)

- Nascent concurrency and weak-memory functionality
- Valuable property: “Just enough specification”
  - Assertions in code act as specifications!
  - Can provide additional specifications in “verification driver” code
- Verified rcu_dereference() and rcu_assign_pointer()
  - Daniel Kroening, Oxford
- Verified Tiny RCU
  - But when I say “Tiny”, I really do mean ***tiny!!!***
- Work in progress: Verify substantial portion of Tree RCU
  - Lihao Liang, Oxford
- Conclusion: Promising, especially if SAT progress continues

Kroening, Clarke, and Lerda, “A tool for checking ANSI-C programs”, *Tools and Algorithms for the Construction and Analysis of Systems, 2004,* pp. 168-176.

Scorecard For Linux-Kernel C Code (Incomplete)

	Promela	PPCMEM	Herd	CBMC
(1) Automated				
(2) Handle environment	(MM)		(MM)	(MM)
(3) Low overhead				SAT?
(4) Map to source code				
(5) Modest input				
(6) Relevant bugs	???	???	???	???
Paul McKenney's first use	1993	2011	2014	2015

Promela MM: Only SC: Weak memory must be implemented in model
Herd MM: Some PowerPC and ARM corner-case issues
CBMC MM: Only SC and TSO
Note: All four handle concurrency! (Promela has done so for 25 years!!!)

Scorecard For Linux-Kernel C Code

	Promela	PPCMEM	Herd	CBMC	Test
(1) Automated					
(2) Handle environment	(MM)		(MM)	(MM)	
(3) Low overhead				SAT?	
(4) Map to source code					
(5) Modest input					
(6) Relevant bugs	???	???	???	???	
Paul McKenney's first use	1993	2011	2014	2015	1973

Scorecard For Linux-Kernel C Code

	Promela	PPCMEM	Herd	CBMC	Test
(1) Automated					
(2) Handle environment	(MM)		(MM)	(MM)	
(3) Low overhead				SAT?	
(4) Map to source code					
(5) Modest input					
(6) Relevant bugs	???	???	???	???	
Paul McKenney's first use	1993	2011	2014	2015	1973

“Annoyance is the feeling of naïvete leaving your mindset.”

Scorecard For Linux-Kernel C Code

	Promela	PPCMEM	Herd	CBMC	Test
(1) Automated					
(2) Handle environment	(MM)		(MM)	(MM)	
(3) Low overhead				SAT?	
(4) Map to source code					
(5) Modest input					
(6) Relevant bugs	???	???	???	???	
Paul McKenney's first use	1993	2011	2014	2015	1973

So why do anything other than testing?

Scorecard For Linux-Kernel C Code

	Promela	PPCMEM	Herd	CBMC	Test
(1) Automated					
(2) Handle environment	(MM)		(MM)	(MM)	
(3) Low overhead				SAT?	
(4) Map to source code					
(5) Modest input					
(6) Relevant bugs	???	???	???	???	
Paul McKenney's first use	1993	2011	2014	2015	1973

So why do anything other than testing?

- Low-probability bugs can require expensive testing regimen
- Large installed base will encounter low-probability bugs
- Safety-criitcal applications are sensitive to low-probability bugs

Other Possible Approaches

- By-hand formalizations and proofs
  - Stern: Semi-formal proof of URCU (2012 IEEE TPDS)
  - Gotsman: Separation-logic RCU semantics (2013 ESOP)
  - Tasserotti et al.: Formal proof of URCU linked list: (2015 PLDI)
  - Excellent work, but not useful for regression testing
- seL4 tooling: Lacks support for concurrency and RCU idioms
  - Might be applicable to Tiny RCU callback handling
  - Impressive work nevertheless!!!
  - (See Gerwin Klein's talk from Monday and Gernot Heiser's upcoming talk.)
- Apply Peter O'Hearn's Infer to the Linux kernel
- Mark Batty et al.: Analyze C11/C++11 memory model
- Jean Pichot-Pharabod: Out-of-thin-air value solution
- Ahmed et al.: Mutations for RCU validation
  - Found numerous holes, one of which was hiding a real bug
  - Upcoming work: Apply mutation to other code bases as well
  - **Conclusion**: Investments in testing still pay off

Challenges And Plan

Challenges

- Find bug in rcu_preempt_offline_tasks()
  - Note: No practical impact because this function has been removed
  - http://paulmck.livejournal.com/37782.html
- Find bug in RCU_NO_HZ_FULL_SYSIDLE
  - http://paulmck.livejournal.com/38016.html
- Find bug in RCU linked-list use cases
  - http://paulmck.livejournal.com/39793.html
- Find lost wakeup bug in the Linux kernel (or maybe qemu)
  - Heavy rcutorture testing with CPU hotplug on two-socket system
  - Detailed repeat-by: https://lkml.org/lkml/2016/3/28/214
  - Can you find this before we do?
- Find any other bug in popular open-source software
  - A verification researcher has provoked a SEGV in Linux-kernel RCU

Plan

- Add formal verification to RCU's Linux-kernel regression test by the end of 2016

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.

Discussion

To Probe Deeper (RCU)

- https://queue.acm.org/detail.cfm?id=2488549
  - "Structured Deferral: Synchronization via Procrastination" (also in July 2013 CACM)
- http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.159 and http://www.computer.org/cms/Computer.org/dl/trans/td/2012/02/extras/ttd2012020375s.pdf
  - "User-Level Implementations of Read-Copy Update"
- git://lttng.org/userspace-rcu.git (User-space RCU git tree)
- http://people.csail.mit.edu/nickolai/papers/clements-bonsai.pdf
  - Applying RCU and weighted-balance tree to Linux mmap_sem.
- http://www.usenix.org/event/atc11/tech/final_files/Triplett.pdf
  - RCU-protected resizable hash tables, both in kernel and user space
- http://www.usenix.org/event/hotpar11/tech/final_files/Howard.pdf
  - Combining RCU and software transactional memory
- http://wiki.cs.pdx.edu/rp/: Relativistic programming, a generalization of RCU
- http://lwn.net/Articles/262464/, http://lwn.net/Articles/263130/, http://lwn.net/Articles/264090/
  - "What is RCU?" Series
- http://www.rdrop.com/users/paulmck/RCU/RCUdissertation.2004.07.14e1.pdf
  - RCU motivation, implementations, usage patterns, performance (micro+sys)
- http://www.livejournal.com/users/james_morris/2153.html
  - System-level performance for SELinux workload: >500x improvement
- http://www.rdrop.com/users/paulmck/RCU/hart_ipdps06.pdf
  - Comparison of RCU and NBS (later appeared in JPDC)
- http://doi.acm.org/10.1145/1400097.1400099
  - History of RCU in Linux (Linux changed RCU more than vice versa)
- http://read.seas.harvard.edu/cs261/2011/rcu.html
  - Harvard University class notes on RCU (Courtesy of Eddie Koher)
- http://www.rdrop.com/users/paulmck/RCU/ (More RCU information)

To Probe Deeper (1/5)

- Hash tables:
  - http://kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook-e1.html Chapter 10
- Split counters:
  - http://kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html Chapter 5
  - http://events.linuxfoundation.org/sites/events/files/slides/BareMetal.2014.03.09a.pdf
- Perfect partitioning
  - Candide et al: “Dynamo: Amazon's highly available key-value store”
    - http://doi.acm.org/10.1145/1323293.1294281
  - McKenney: “Is Parallel Programming Hard, And, If So, What Can You Do About It?”
    - http://kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html Section 6.5
  - McKenney: “Retrofitted Parallelism Considered Grossly Suboptimal”
    - Embarrassing parallelism vs. humiliating parallelism
    - https://www.usenix.org/conference/hotpar12/retro%EF%AC%81tted-parallelism-considered-grossly-sub-optimal
  - McKenney et al: “Experience With an Efficient Parallel Kernel Memory Allocator”
    - http://www.rdrop.com/users/paulmck/scalability/paper/mpalloc.pdf
  - Bonwick et al: “Magazines and Vmem: Extending the Slab Allocator to Many CPUs and Arbitrary Resources”
    - http://static.usenix.org/event/usenix01/full_papers/bonwick/bonwick_html/
  - Turner et al: “PerCPU Atomics”
    - http://www.linuxplumbersconf.org/2013/ocw//system/presentations/1695/original/LPC%20-%20PerCpu%20Atomics.pdf

To Probe Deeper (2/5)

- Stream-based applications:
  - Sutton: “Concurrent Programming With The Disruptor”
    - http://www.youtube.com/watch?v=UvE389P6Er4
    - http://lca2013.linux.org.au/schedule/30168/view_talk
  - Thompson: “Mechanical Sympathy”
    - http://mechanical-sympathy.blogspot.com/
- Read-only traversal to update location
  - Arcangeli et al: “Using Read-Copy-Update Techniques for System V IPC in the Linux 2.5 Kernel”
    - https://www.usenix.org/legacy/events/usenix03/tech/freenix03/full_papers/arcangeli/arcangeli_html/index.html
  - Corbet: “Dcache scalability and RCU-walk”
    - https://lwn.net/Articles/419811/
  - Xu: “bridge: Add core IGMP snooping support”
    - http://kerneltrap.com/mailarchive/linux-netdev/2010/2/26/6270589
  - Triplett et al., “Resizable, Scalable, Concurrent Hash Tables via Relativistic Programming”
    - http://www.usenix.org/event/atc11/tech/final_files/Triplett.pdf
  - Howard: “A Relativistic Enhancement to Software Transactional Memory”
    - http://www.usenix.org/event/hotpar11/tech/final_files/Howard.pdf
  - McKenney et al: “URCU-Protected Hash Tables”
    - http://lwn.net/Articles/573431/

To Probe Deeper (3/5)

- Hardware lock elision: Overviews
  - Kleen: “Scaling Existing Lock-based Applications with Lock Elision”
    - http://queue.acm.org/detail.cfm?id=2579227
- Hardware lock elision: Hardware description
  - POWER ISA Version 2.07
    - http://www.power.org/documentation/power-isa-version-2-07/
  - Intel® 64 and IA-32 Architectures Software Developer Manuals
    - http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html
  - Jacobi et al: “Transactional Memory Architecture and Implementation for IBM System z”
    - http://www.microsymposia.org/micro45/talks-posters/3-jacobi-presentation.pdf
- Hardware lock elision: Evaluations
  - http://pcl.intel-research.net/publications/SC13-TSX.pdf
  - http://kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html Section 16.3
- Hardware lock elision: Need for weak atomicity
  - Herlihy et al: “Software Transactional Memory for Dynamic-Sized Data Structures”
    - http://research.sun.com/scalable/pubs/PODC03.pdf
  - Shavit et al: “Data structures in the multicore age”
    - http://doi.acm.org/10.1145/1897852.1897873
  - Haas et al: “How FIFO is your FIFO queue?”
    - http://dl.acm.org/citation.cfm?id=2414731
  - Gramoli et al: “Democratizing transactional programming”
    - http://doi.acm.org/10.1145/2541883.2541900

To Probe Deeper (4/5)

- RCU
  - Desnoyers et al.: “User-Level Implementations of Read-Copy Update”
    - http://www.rdrop.com/users/paulmck/RCU/urcu-main-accepted.2011.08.30a.pdf
    - http://www.computer.org/cms/Computer.org/dl/trans/td/2012/02/extras/ttd2012020375s.pdf
  - McKenney et al.: “RCU Usage In the Linux Kernel: One Decade Later”
    - http://rdrop.com/users/paulmck/techreports/survey.2012.09.17a.pdf
    - http://rdrop.com/users/paulmck/techreports/RCUUsage.2013.02.24a.pdf
  - McKenney: “Structured deferral: synchronization via procrastination”
    - http://doi.acm.org/10.1145/2483852.2483867
  - McKenney et al.: “User-space RCU” https://lwn.net/Articles/573424/
- Possible future additions
  - Boyd-Wickizer: “Optimizing Communications Bottlenecks in Multiprocessor Operating Systems Kernels”
    - http://pdos.csail.mit.edu/papers/sbw-phd-thesis.pdf
  - Clements et al: “The Scalable Commutativity Rule: Designing Scalable Software for Multicore Processors”
    - http://www.read.seas.harvard.edu/~kohler/pubs/clements13scalable.pdf
  - McKenney: “N4037: Non-Transactional Implementation of Atomic Tree Move”
    - http://www.rdrop.com/users/paulmck/scalability/paper/AtomicTreeMove.2014.05.26a.pdf
  - McKenney: “C++ Memory Model Meets High-Update-Rate Data Structures”
    - http://www2.rdrop.com/users/paulmck/RCU/C++Updates.2014.09.11a.pdf

To Probe Deeper (5/5)

- RCU theory and semantics, academic contributions (partial list)
  - Gamsa et al., “Tornado: Maximizing Locality and Concurrency in a Shared Memory Multiprocessor Operating System”
    - http://www.usenix.org/events/osdi99/full_papers/gamsa/gamsa.pdf
  - McKenney, “Exploiting Deferred Destruction: An Analysis of RCU Techniques”
    - http://www.rdrop.com/users/paulmck/RCU/RCUdissertation.2004.07.14e1.pdf
  - Hart, “Applying Lock-free Techniques to the Linux Kernel”
    - http://www.cs.toronto.edu/~tomhart/masters_thesis.html
  - Olsson et al., “TRASH: A dynamic LC-trie and hash data structure”
    - http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4281239
  - Desnoyers, “Low-Impact Operating System Tracing”
    - http://www.lttng.org/pub/thesis/desnoyers-dissertation-2009-12.pdf
  - Dalton, “The Design and Implementation of Dynamic Information Flow Tracking ...”
    - http://csl.stanford.edu/~christos/publications/2009.michael_dalton.phd_thesis.pdf
  - Gotsman et al., “Verifying Highly Concurrent Algorithms with Grace (extended version)”
    - http://software.imdea.org/~gotsman/papers/recycling-esop13-ext.pdf
  - Liu et al., “Mindicators: A Scalable Approach to Quiescence”
    - http://dx.doi.org/10.1109/ICDCS.2013.39
  - Tu et al., “Speedy Transactions in Multicore In-memory Databases”
    - http://doi.acm.org/10.1145/2517349.2522713
  - Arbel et al., “Concurrent Updates with RCU: Search Tree as an Example”
    - http://www.cs.technion.ac.il/~mayaarl/podc047f.pdf