Searching for Data Races with Support for RCU

Anton Volkov, Pavel Andrianov, Vadim Mutilin

What is Read-copy Update Fundamentally

1. Appeared in the Linux kernel in October of 2002
2. Provides read-write synchronization with efficient reads

See Paul E. McKenney, Jonathan Walpole
https://lwn.net/Articles/262464/

Example of RCU

Writer

old_p = **deref**(**rcu_p**)

new_p = malloc

updating new_p

assign(**rcu_p**, new_p)

synchronize()

free(old_p)

Reader

read_lock()

p = **deref**(**rcu_p**)

reading p

read_unlock()

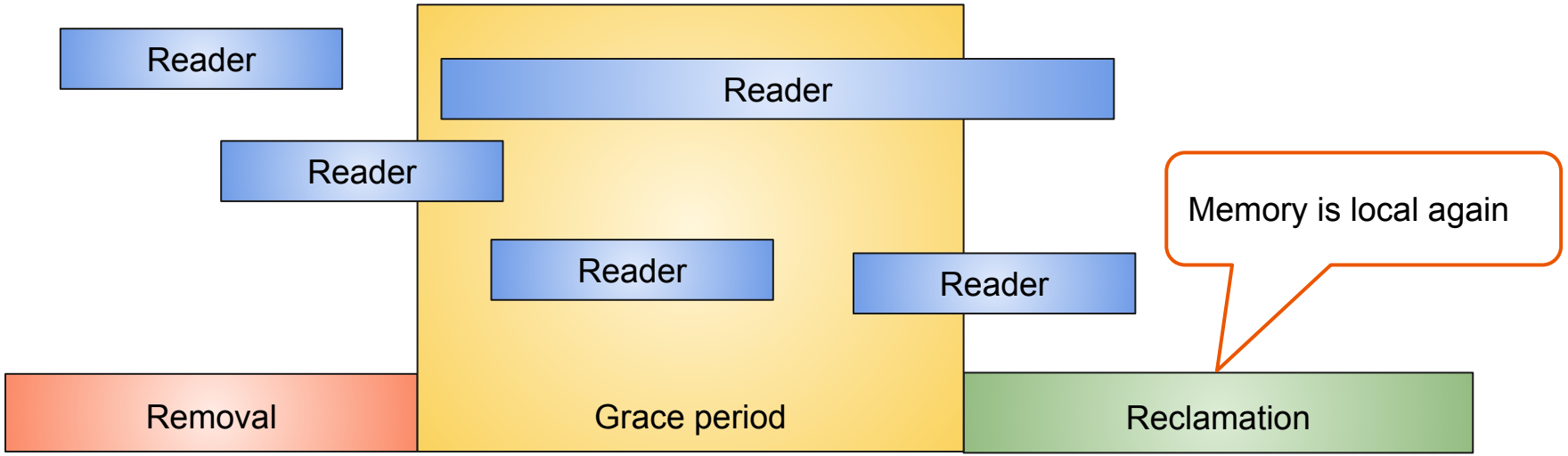
Example of RCU

	Writer	Reader
	old_p = **deref**(**rcu_p**)	**read_lock**()
	new_p = malloc	p = **deref**(**rcu_p**)
	updating new_p	reading p
Removal	**assign**(**rcu_p**, new_p)	**read_unlock**()
	synchronize()	
	free(old_p)	

Example of RCU

	Writer	Reader
	old_p = **deref**(**rcu_p**)	**read_lock**()
	new_p = malloc	p = **deref**(**rcu_p**)
	updating new_p	reading p
Removal	**assign**(**rcu_p**, new_p)	**read_unlock**()
	synchronize()	
Reclamation	free(old_p)	

Removal & Reclamation



Example of RCU

Writer

old_p = **deref**(**rcu_p**)

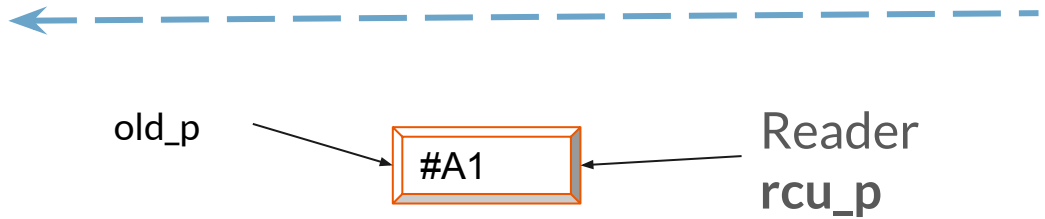
new_p = malloc

updating new_p

assign(**rcu_p**, new_p)

synchronize()

free(old_p)



Example of RCU

Writer

old_p = **deref**(**rcu_p**)

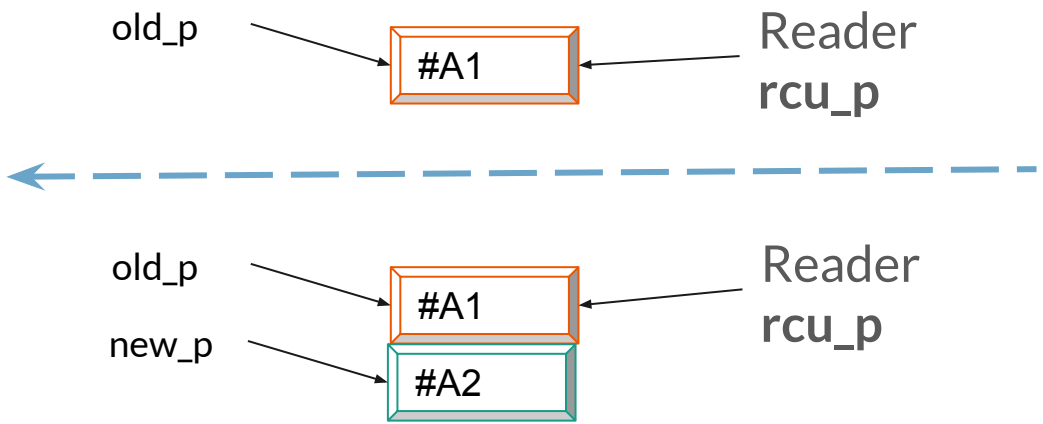
new_p = malloc

updating new_p

assign(**rcu_p**, new_p)

synchronize()

free(old_p)



Example of RCU

Writer

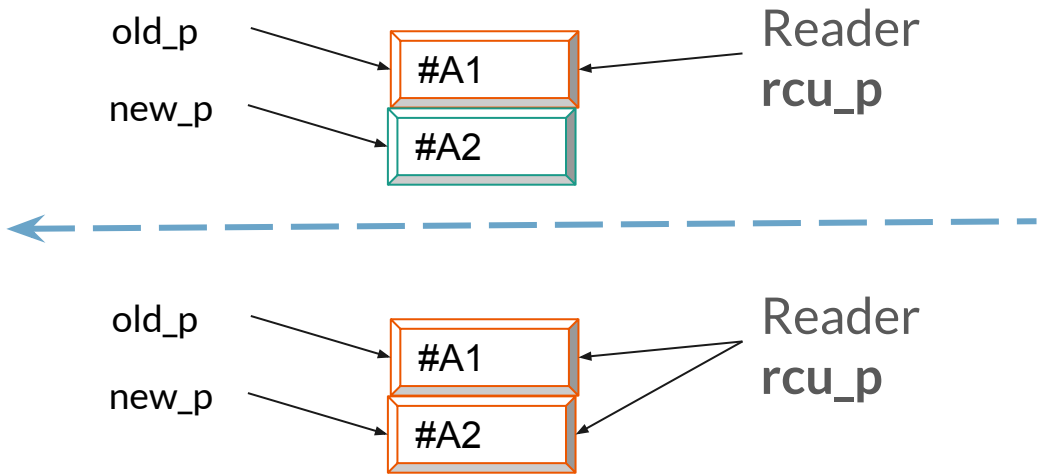
...

updating new_p

assign(**rcu_p**, new_p)

synchronize()

free(old_p)



Example of RCU

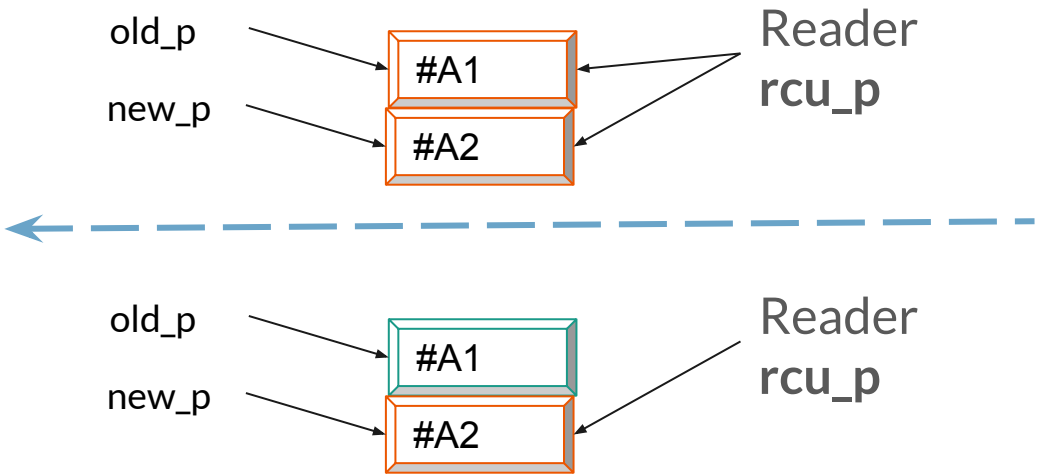
Writer

...

assign(**rcu_p**, new_p)

synchronize()

free(old_p)



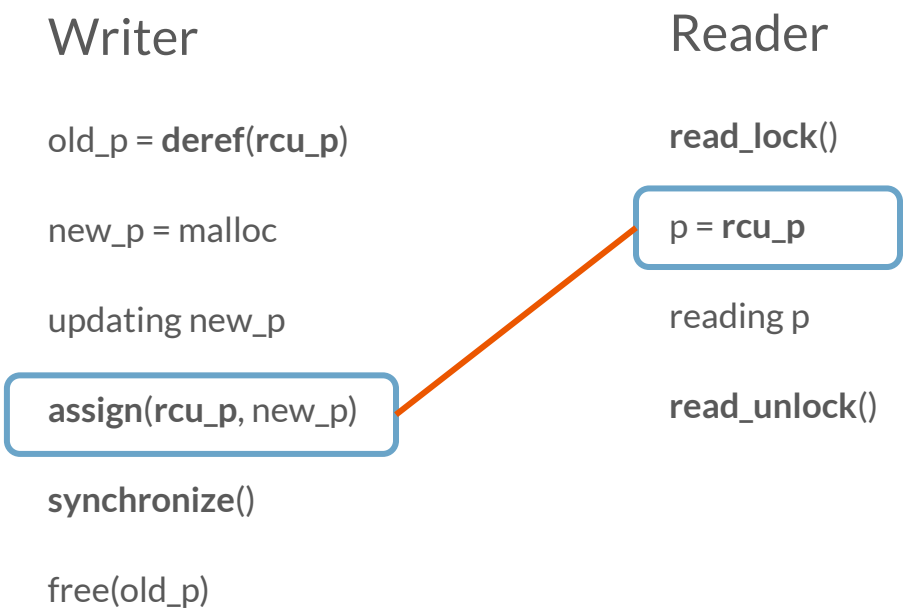
Definition of Data Race

Data race is two or more simultaneous accesses to the same memory one of which is write

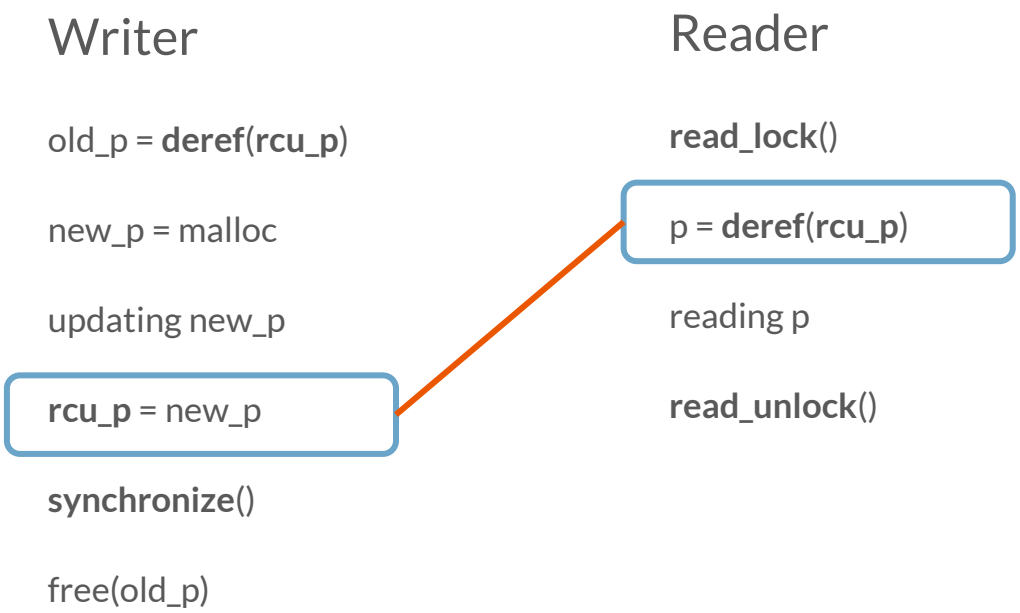
Claims:

1) same shared memory
2) one of the accesses is write
3) appear simultaneously

RCU Data Races - Holding Locks (1)



RCU Data Races - Holding Locks (2)



RCU Data Races - Locality

Writer

old_p = **deref**(**rcu_p**)

new_p = malloc

updating new_p

assign(**rcu_p**, new_p)

~~**synchronize**()~~

free(old_p)

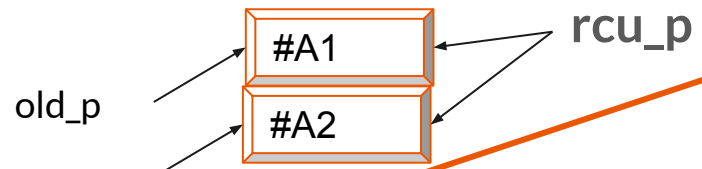
Reader

read_lock()

p = **deref**(**rcu_p**)

reading p

read_unlock()



Searching for Data Races

CPALockator key ideas

- abstract transitions
- compatibility of states

CPALockator - Data Race

Claims:

1) same shared memory

2) one of the accesses is write

3) appear simultaneously

CPALockator - Data Race

Claims:

1) same shared memory

2) one of the accesses is write

3) appear simultaneously

Consider abstract transitions a, b
state(a) - abstract state
edge(a) - abstract edge

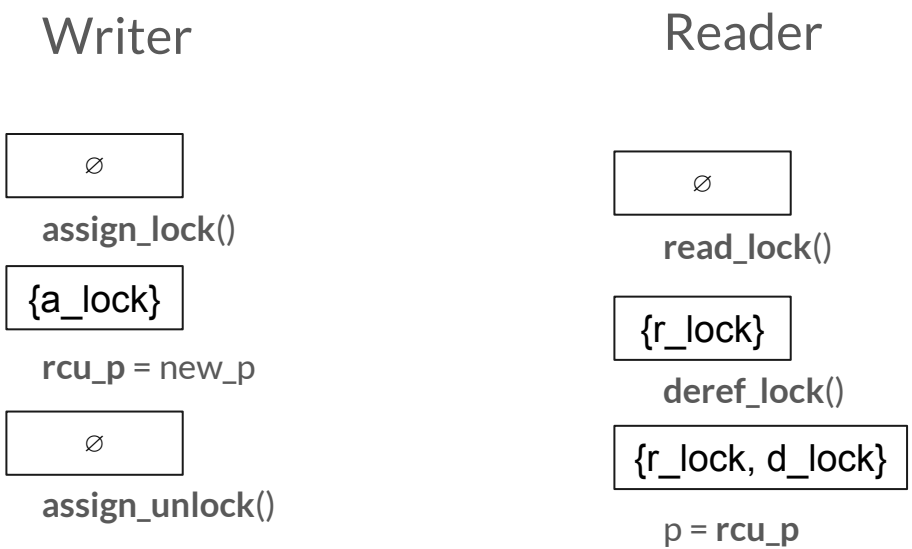
1.a) memory(edge(a)) == memory(edge(b))

1.b) isShared(memory(edge(a)))

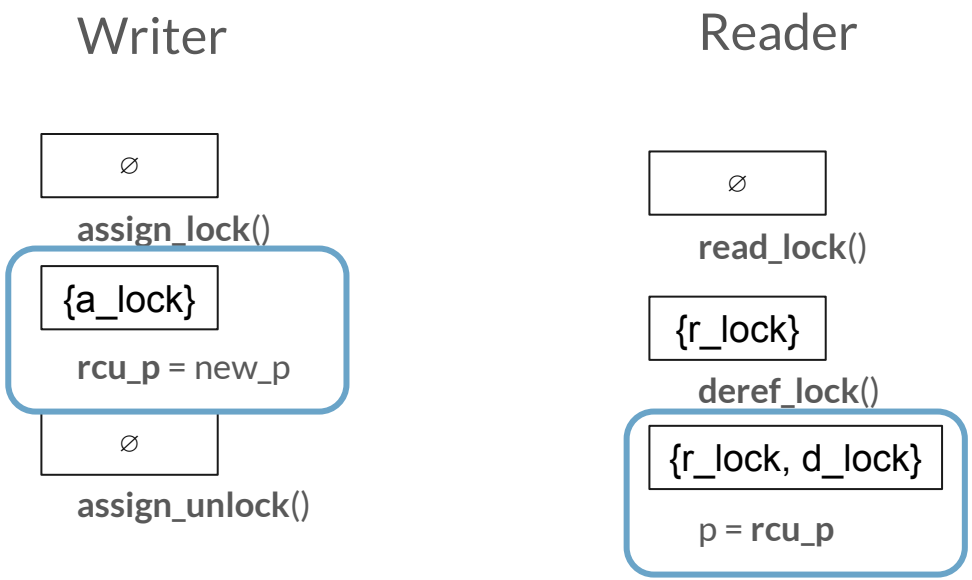
2) access(edge(a)) ==WRITE ||
access(edge(b)) ==WRITE

3) compatible(state(a),state(b))

CPALockator - Tracking Locks

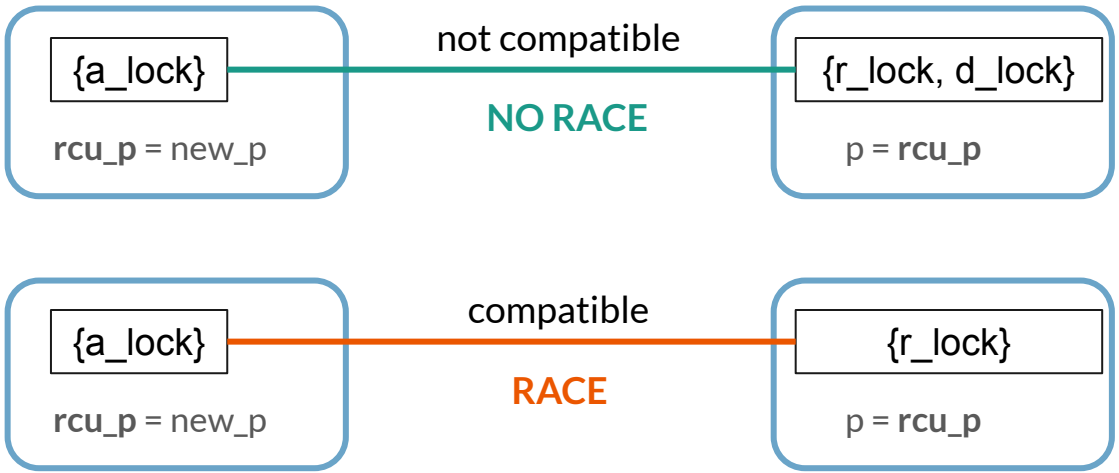


CPALockator - Tracking Locks



CPALockator - Compatibility

CPALockator - Compatibility



CPALockator - Data Race

Claims:

1) same shared memory

2) one of the accesses is write

3) appear simultaneously

Consider abstract transitions a, b

1.a) memory(edge(a)) == memory(edge(b))

1.b) isShared(memory(edge(a)))

2) access(edge(a)) ==WRITE ||
access(edge(b)) ==WRITE

3) compatible(state(a),state(b))

RCUCPA - Tracking Locality

Writer

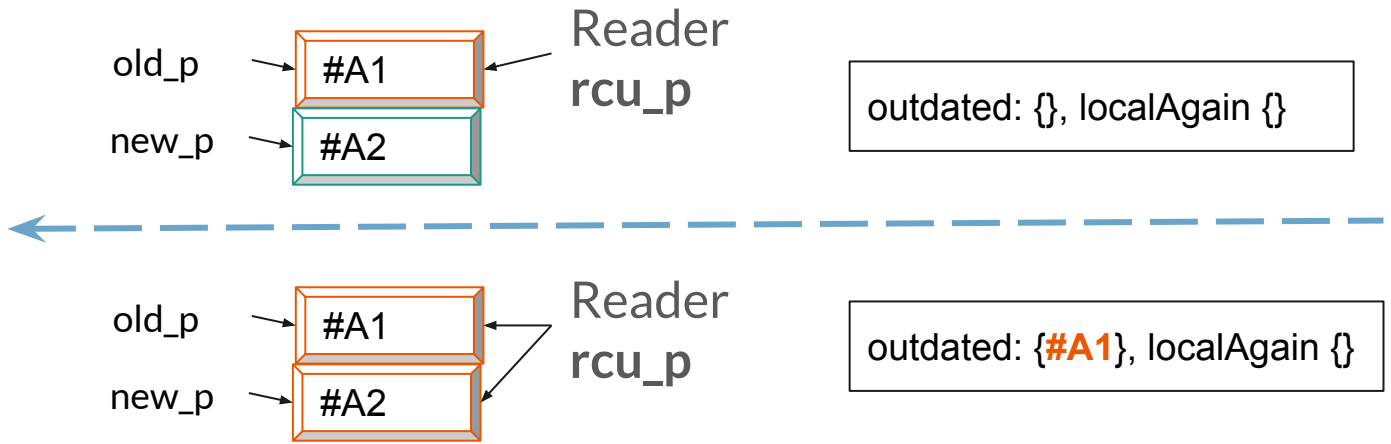
...

updating new_p

assign(**rcu_p**, new_p)

synchronize()

free(old_p)



RCUCPA - Tracking Locality

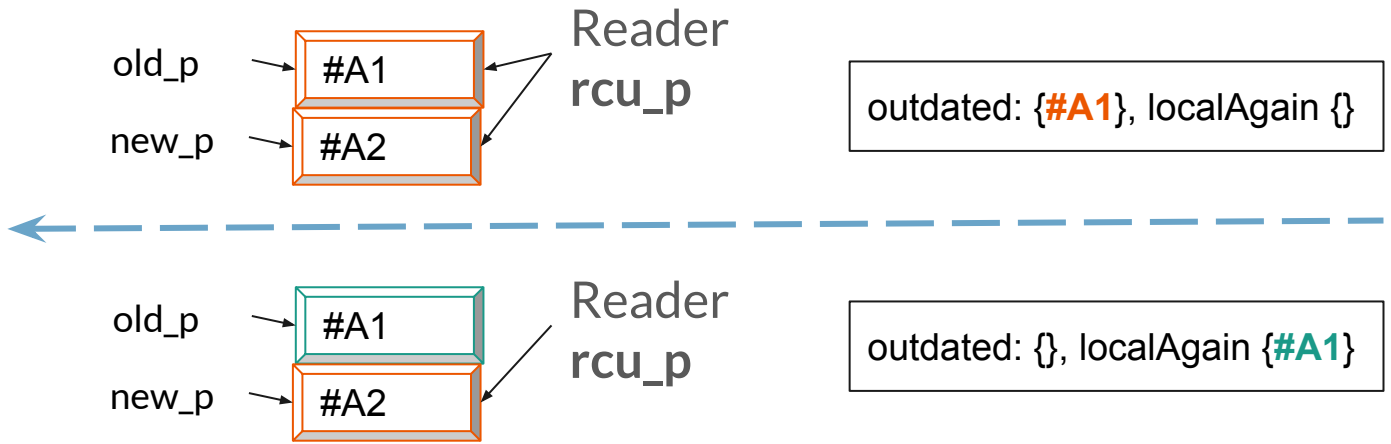
Writer

...

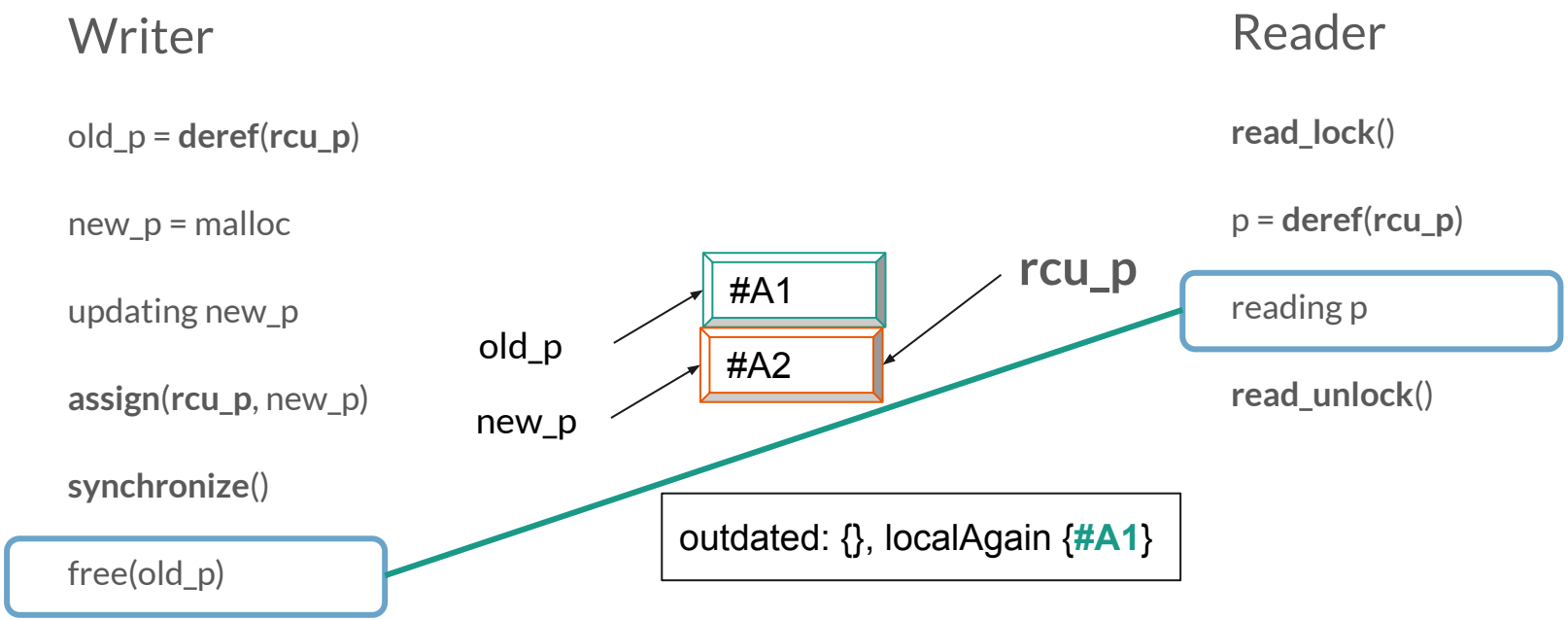
assign(**rcu_p**, new_p)

synchronize()

free(old_p)



RCU Data Races - Locality



RCUCPA - isShared

outdated: {}, localAgain {**#A1**}

isShared(#A1) = false

Consider abstract transitions a, b

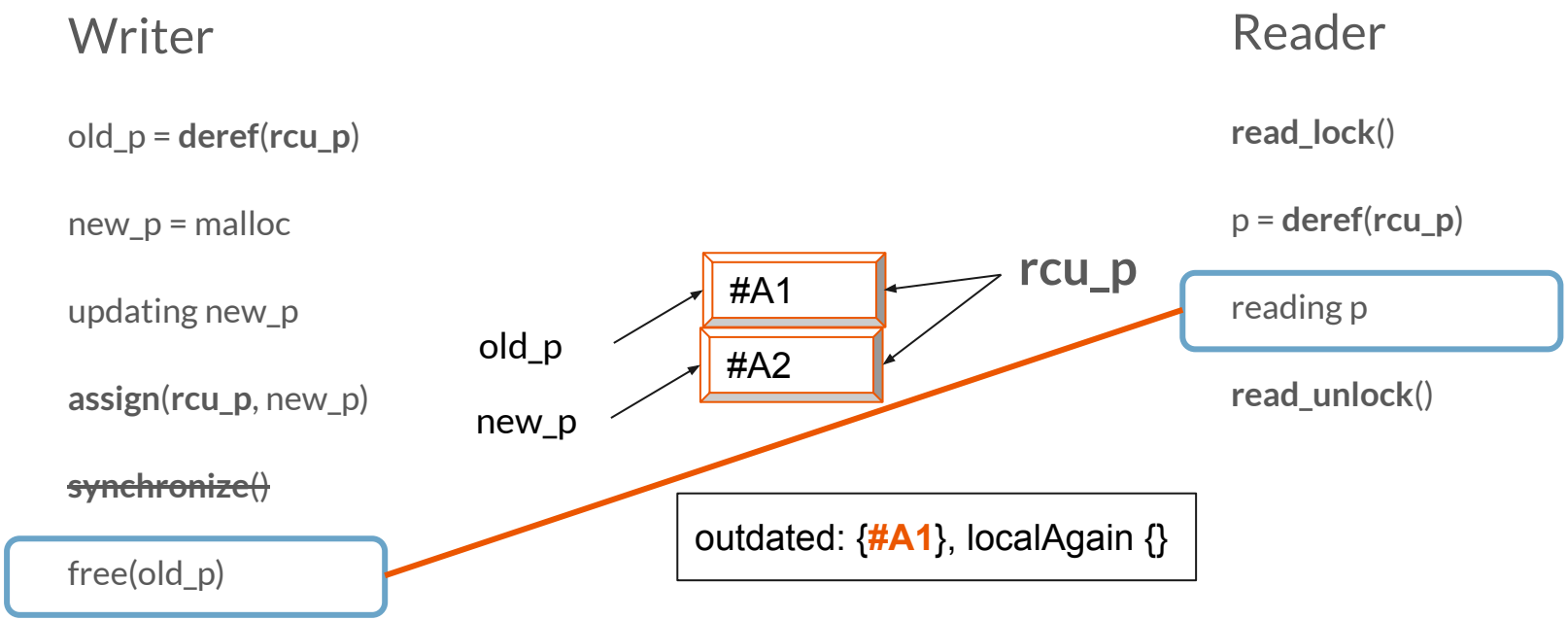
1.a) memory(edge(a)) == memory(edge(b))

1.b) isShared(memory(edge(a)))

2) access(edge(a)) ==WRITE ||
access(edge(b)) ==WRITE

3) compatible(state(a),state(b))

RCU Data Races - Locality



RCUCPA - isShared

outdated: {**#A1**}, localAgain {}

isShared(#A1) = true

Consider abstract transitions a, b

1.a) memory(edge(a)) == memory(edge(b))

1.b) isShared(memory(edge(a)))

2) access(edge(a)) ==WRITE ||
access(edge(b)) ==WRITE

3) compatible(state(a),state(b))

Implementation

- Originally developed by Anton Volkov as Master thesis
- Refactored in *CPALockator-combat-mode* branch by Pavel Andrianov

Results on Linux kernel drivers 4.2.6.

Total		65	
Fail before CPALockator		19	
False		0	
True	not covered by env model	29	27
	no race		2
Timeout		17	

Searching for Known Bugs

Taken from the analysis of known errors in te Linux kernel in 2014 (work done by ISPRAS)

module (commit)	Before fix	After fix
net/xfrm/xfrm_user.ko (21ee543edc)	False	True
net/core/drop_monitor.ko (3885ca785a)	False	False*
drivers/net/usb/cdc_mbim.ko (4f4178f3bb)	Timeout	Timeout

*Non target bug found

Thanks

- Anton Volkov
- Pavel Andrianov
- Vadim Mutilin

* The research was carried out with funding from the Ministry of Science and Higher Education of the Russian Federation (the project unique identifier is RFMEFI60719X0295)