Using RCU Techniques for SysV IPC in Linux 2.5

Andrea Arcangeli, *SuSE*
Mingming Cao, *IBM Beaverton*
Paul E. McKenney, *IBM Beaverton*
Dipankar Sarma, *IBM India SW Lab*

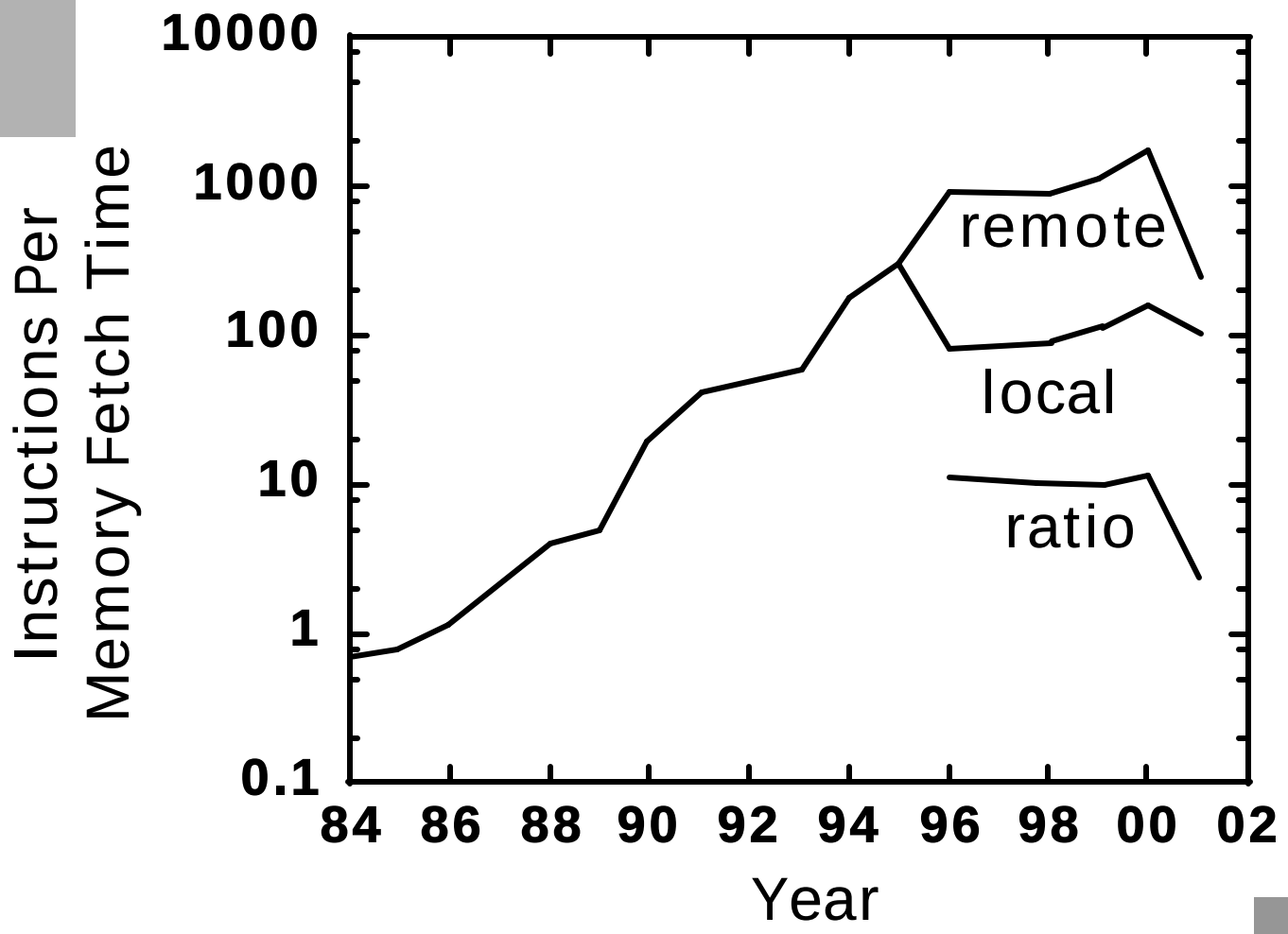
Overview

- *Background on read-copy update (RCU)*
  - Motivation: changing CPU architecture
  - Analogy to reader-writer locking
  - Linked-list example
- Use of RCU techniques for System-V IPC in the Linux kernel
- Concluding remarks

RCU Motivation

- Atomic-operation overhead increasing
  - 700 MHz P3: 55.3 ns (~39 cycles)
  - 1.8 GHz P4: 75.1 ns (~*135* cycles!!!)
- Pipeline flushing increasingly expensive
- Incur cache-miss overhead on top of this!!!
- Why pay these costs for read-mostly data?
  - Use RCU for route caches, config data, etc.

RCU Motivation: Architectural Trends



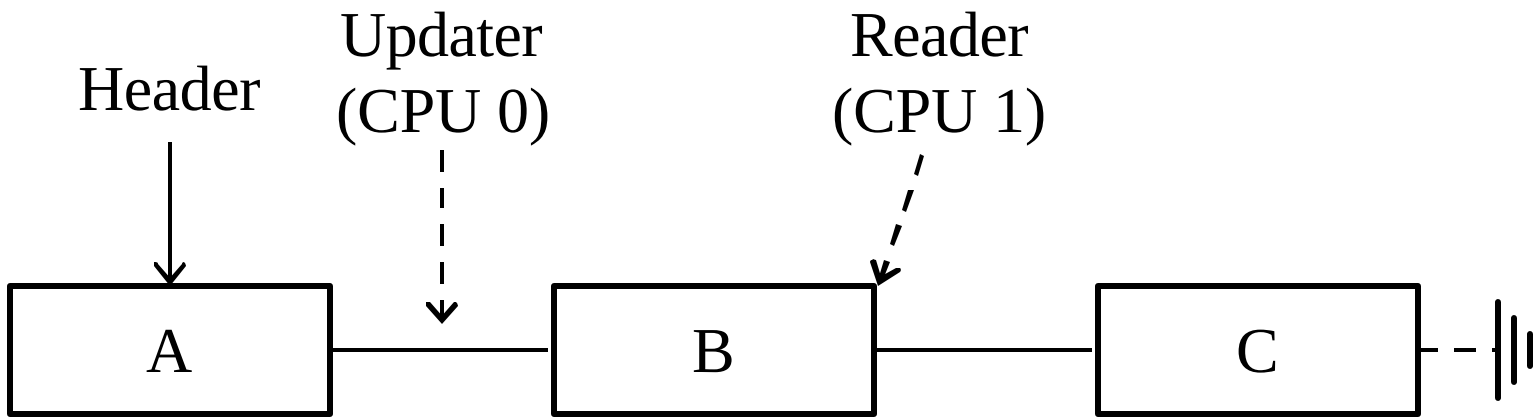
Data from Sequent®/IBM® NUMA-Q®

Reader-Writer-Lock Analogy to RCU

- rwlock_t
- read_lock()
- read_unlock()
- write_lock()
- write_unlock()
- list_add()
- list_del
- list_for_each()
- kfree()

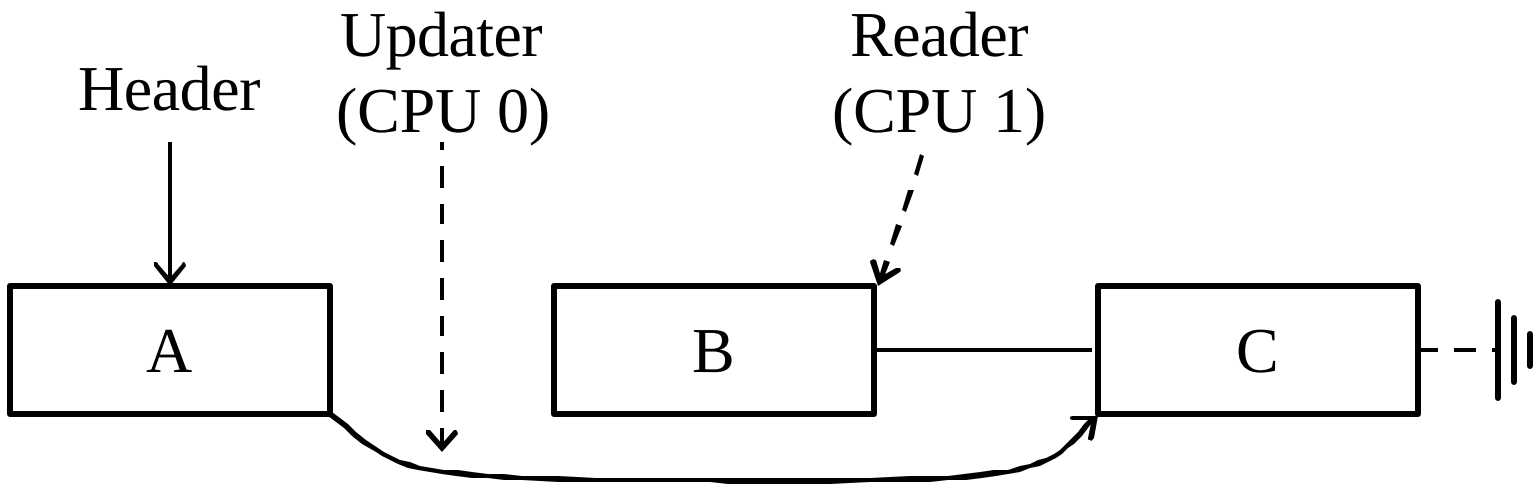
- spinlock_t
- rcu_read_lock() ***
- rcu_read_unlock() ***
- spin_lock()
- spin_unlock()
- list_add_rcu()
- list_del_rcu()
- list_for_each_rcu()
- call_rcu(rhp, kfree, p)

RCU Linked-List Example




```
*predp = p->next;
```

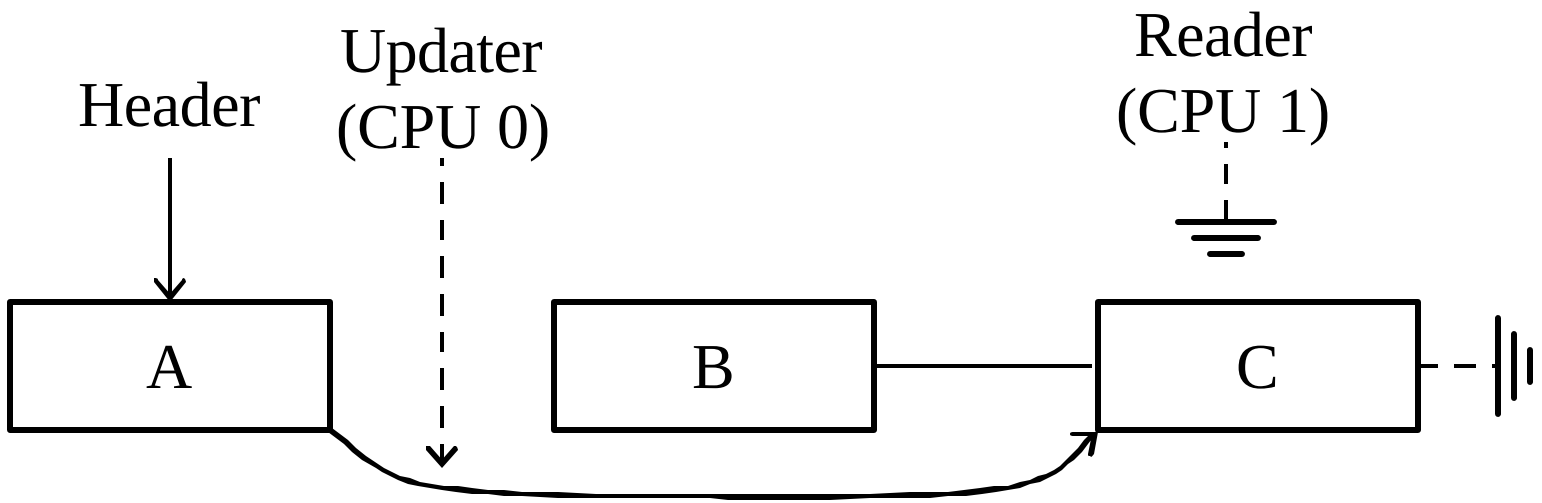
Read-Copy Update Animation




```
call_rcu(&p->rcu_head, kfree, p);
```

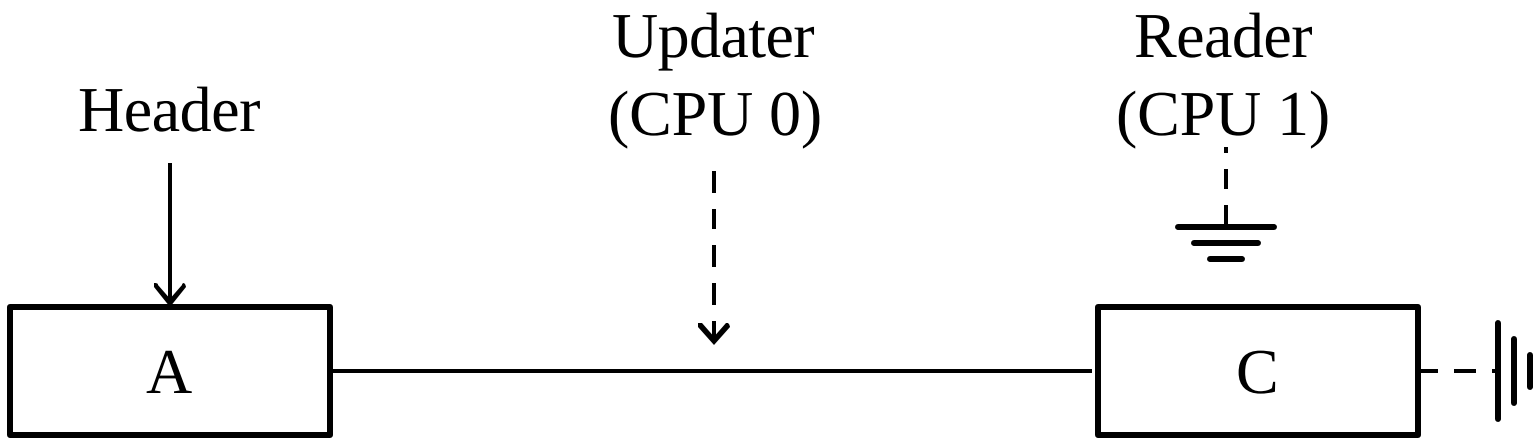
Read-Copy Update Animation

Once each CPU has context switched…




```
kfree(p);
```

Read-Copy Update Animation



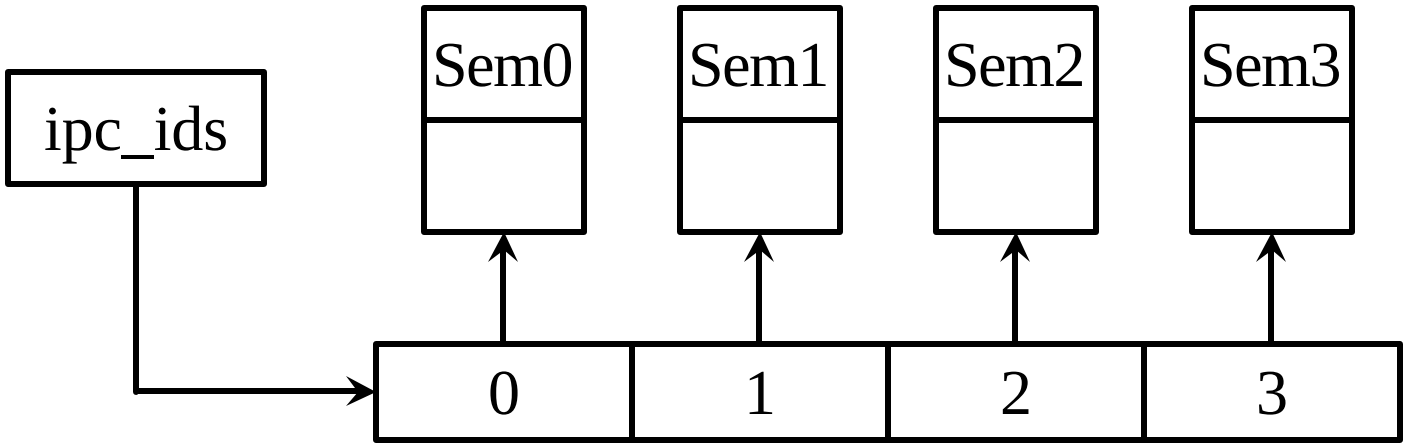
Overview

- Background on read-copy update (RCU)
  - Motivation: changing CPU architecture
  - Analogy to reader-writer locking
  - Linked-list example
- *Use of RCU techniques for System-V IPC in the Linux kernel*
- Concluding remarks

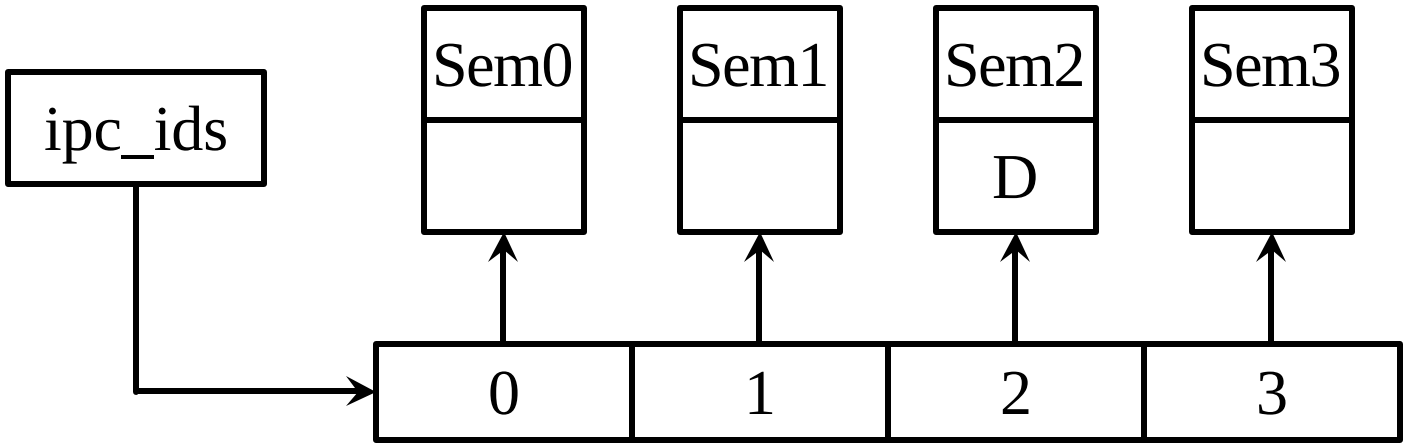
RCU for SysV Semaphores

- Original implementation used global locks
  - High lock contention during database benchmark
- SemID mapping is read mostly
- Excellent candidate for RCU
- Small change: 342 LOC added, 191 LOC deleted, net 151 LOC added
- Large performance gain for microbenchmark
  - Significant for database benchmark

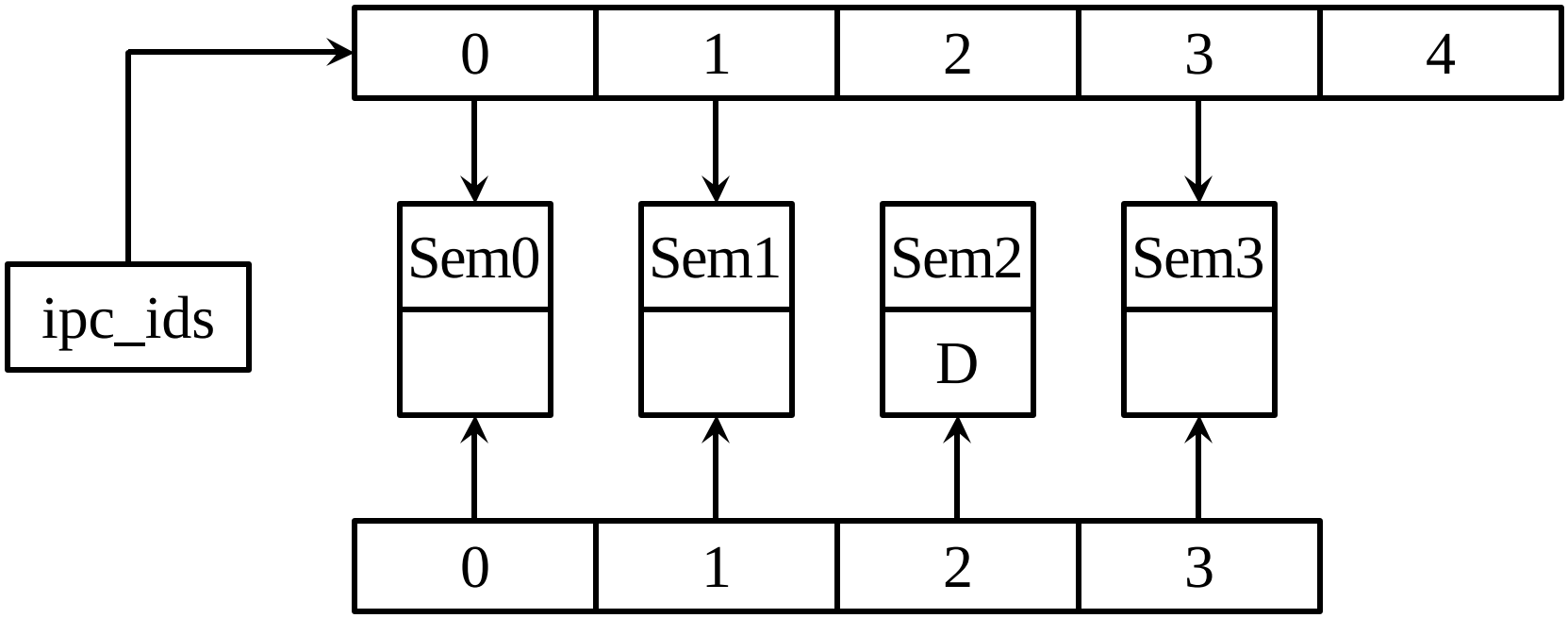
Sema Structures Initial State



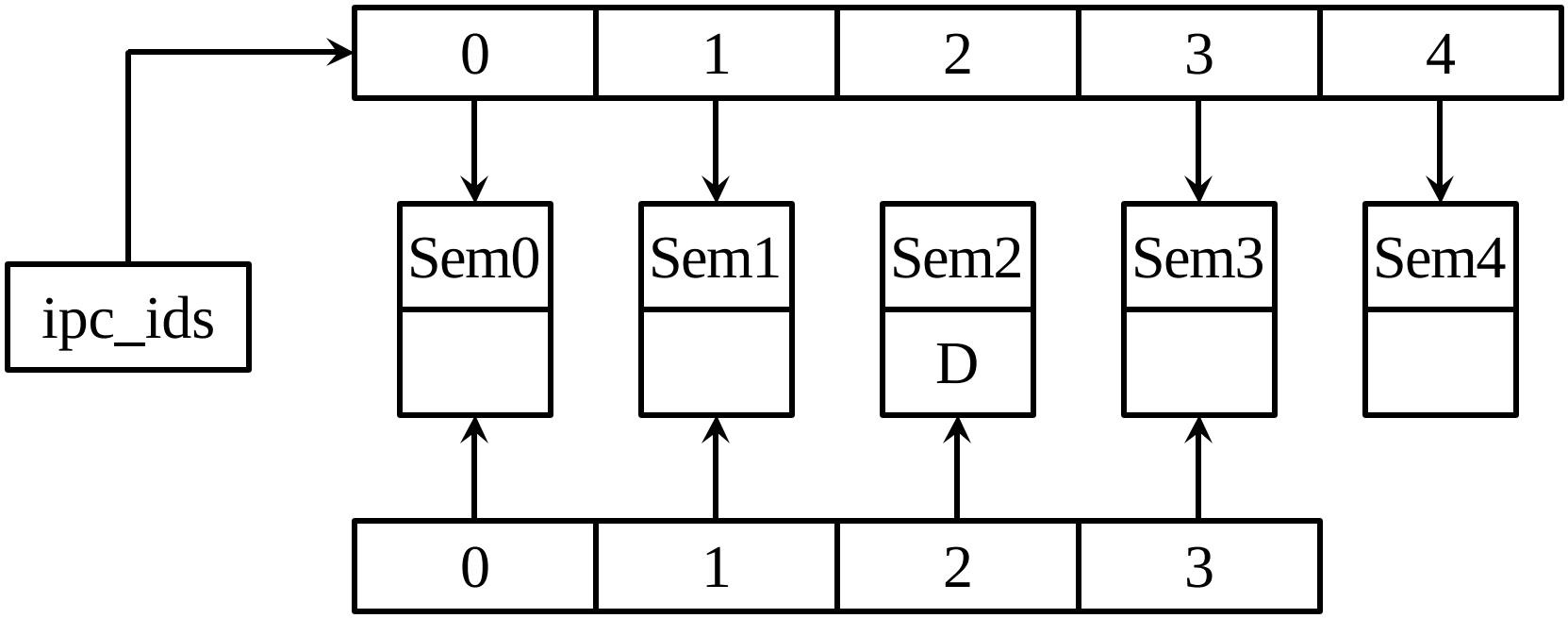
Sema Structures After Sem2 Deletion



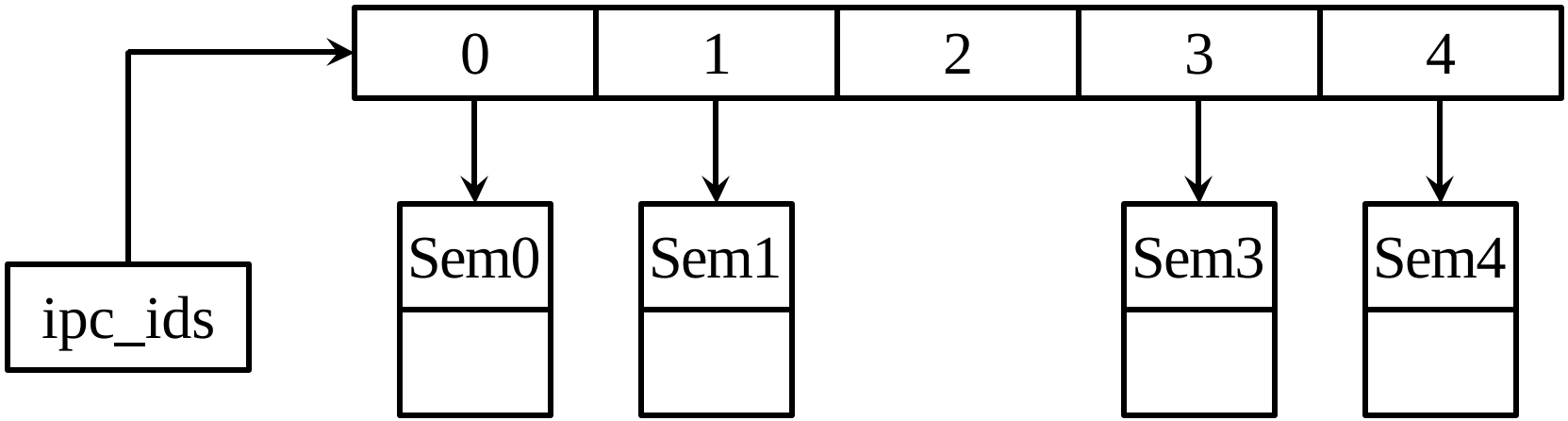
Sema Structures After Growing Array



Sema Structures After Adding Sem4



Sema Structures After Grace Period



Sema Performance Results

Kernel	Run 1 Time (s)	Run 2 Time (s)
2.5.42-mm2	515.1	515.3
2.5.42-mm2+ipc-rcu	46.7	46.7

Kernel	Average TPS	Std. Dev. TPS
2.5.42-mm2	85.0	7.5
2.5.42-mm2+ipc-rcu	89.8	1.0

Overview

- Background on read-copy update (RCU)
  - Motivation: changing CPU architecture
  - Analogy to reader-writer locking
  - Linked-list example
- Use of RCU techniques for System-V IPC in the Linux kernel
- *Concluding remarks*

Concluding Remarks

- RCU can provide significant performance benefits with small code changes
- Future plans:
  - Continue applying RCU to the Linux kernel
  - Prototype user-level RCU implementation

#include <disclaimers.h>

1. The views expressed in this paper are the authors' only, and should not be attributed to SuSE or IBM.
2. IBM, DYNIX/ptx, NUMA-Q, and Sequent are registered trademarks of International Business Machines Corporation in the United States and/or other countries.
3. Linux is a registered trademark of Linus Torvalds.
4. Other company, product, and service names may be trademarks or service marks of others.

Availability

- Read-copy update may be freely used under GPL.
  - http://sourceforge.net/projects/lse/
  - ftp://kernel.org/pub/linux/kernel/2.5
- Other papers available
  - http://www.rdrop.com/users/paulmck