Preemptable Ticket Spinlocks

Improving Consolidated Performance in the Cloud

Jiannan Ouyang, John Lange

Department of Computer Science

University of Pittsburgh

VEE '13

03/17/2013

Motivation

- VM interference in overcommitted environments
  - OS synchronization overhead
  - Lock holder preemption (LHP)

Contributions

- Lock *Waiter* Preemption
  - significance analysis of lock waiter preemption
- Preemptable Ticket Spinlock
  - implementation inside Linux
- Evaluation
  - significant speedup over Linux

Spinlocks

- Basics
  - lock() & unlock()
  - Busy waiting lock
  - generic spinlock: random order, unfair (starvation)
  - ticket spinlock: FIFO order, fair

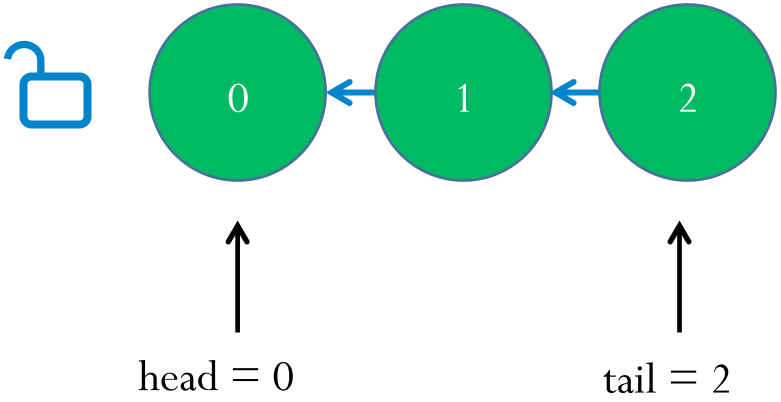
- Designed for fast mutual exclusion
  - busy waiting vs. sleep/wakeup
    - spinlocks for short & fast critical sections (~1us)
  - OS assumptions
    - use spinlocks for short critical section only
    - never preempt a thread holding or waiting a kernel spinlock

Preemption in VMs

- Lock Holder Preemption (LHP)
  - virtualization breaks the OS assumption
  - vCPU holding a lock is unscheduled by VMM
  - preemption prolongs critical section (~1m v.s. ~1us)

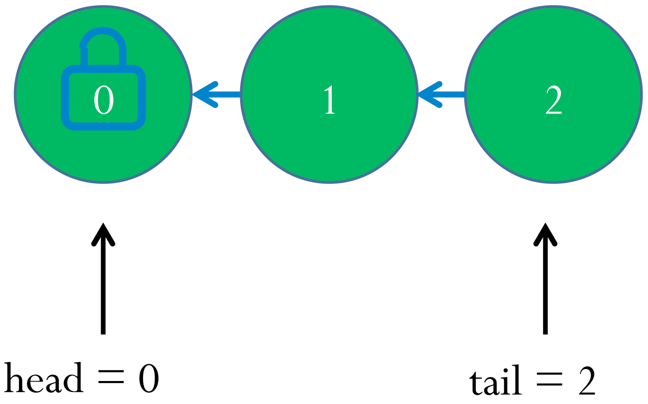
- Proposed Solutions
  - Co-scheduling and variants
  - Hardware-assisted scheme (Pause Loop Exiting)
  - Paravirtual spinlocks

Preemption in Ticket Lock



- 0 a **scheduled** waiter with ticket 0
- 1 a **preempted** waiter with ticket 1

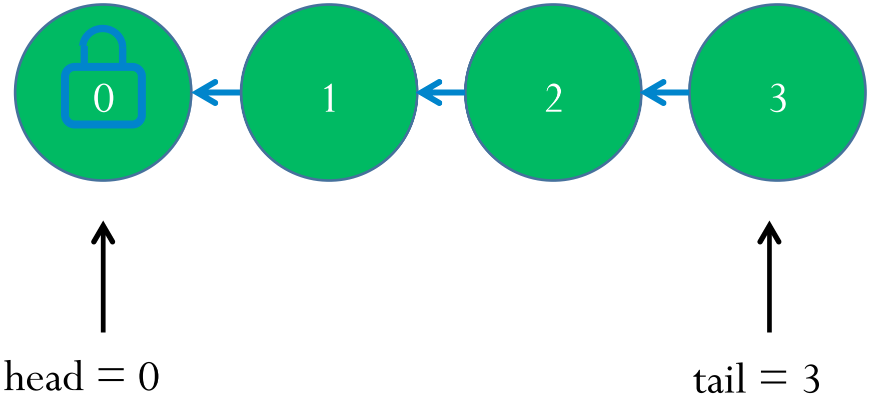
Preemption in Ticket Lock



0 a **scheduled** waiter with ticket 0

1 a **preempted** waiter with ticket 1

Preemption in Ticket Lock



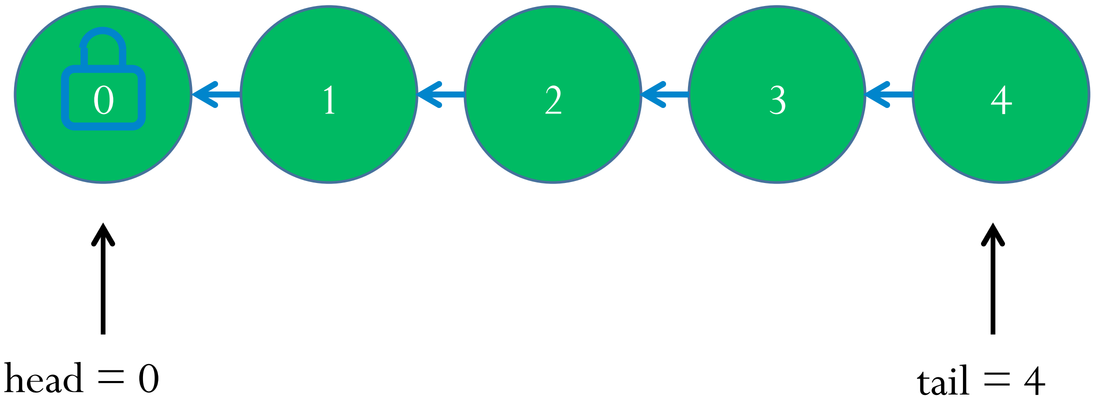
head = 0

tail = 3

🟢 0 a **scheduled** waiter with ticket 0

🔴 1 a **preempted** waiter with ticket 1

Preemption in Ticket Lock



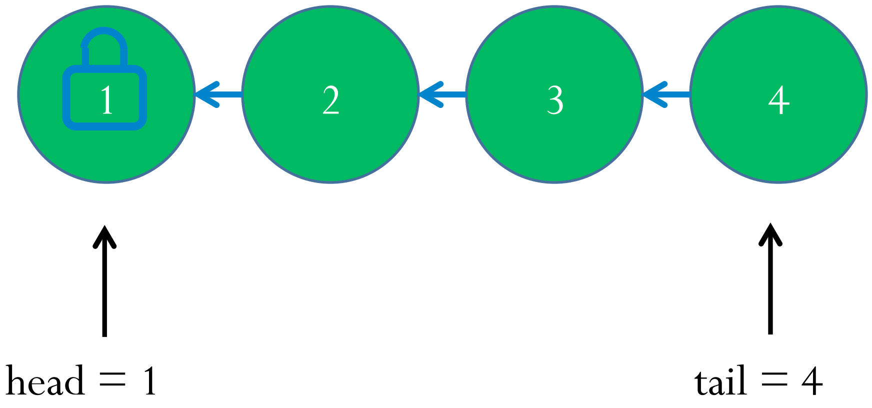
head = 0

tail = 4

🟢 0 a **scheduled** waiter with ticket 0

🔴 1 a **preempted** waiter with ticket 1

Preemption in Ticket Lock

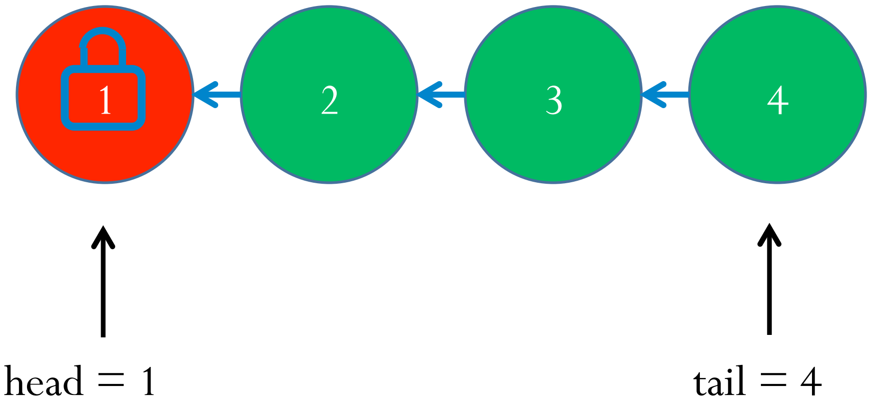


🟩 0 a **scheduled** waiter with ticket 0

🟥 1 a **preempted** waiter with ticket 1

Preemption in Ticket Lock

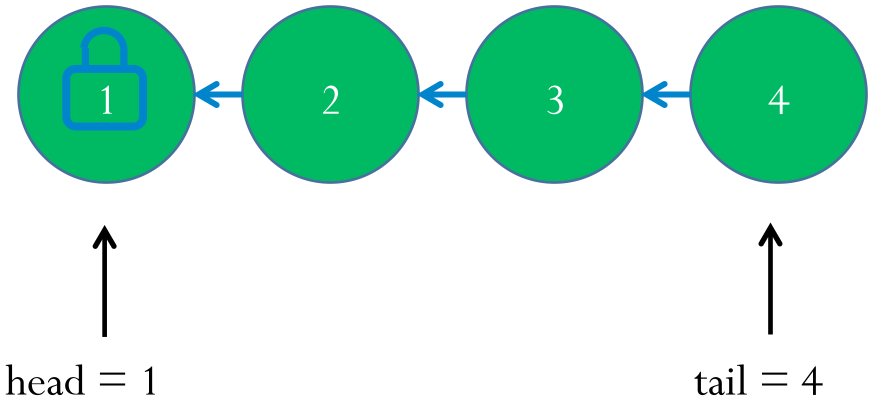
Lock Holder Preemption!



0 a **scheduled** waiter with ticket 0

1 a **preempted** waiter with ticket 1

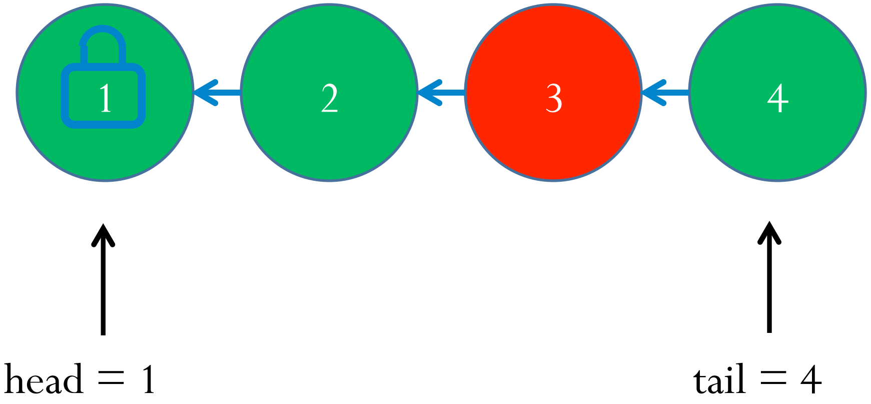
Preemption in Ticket Lock



0 a **scheduled** waiter with ticket 0

1 a **preempted** waiter with ticket 1

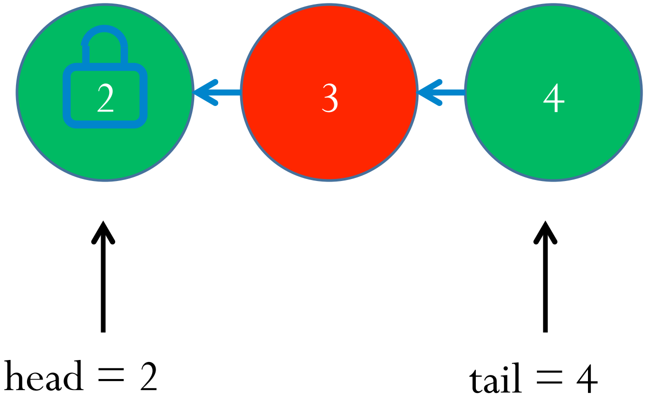
Preemption in Ticket Lock



0 a **scheduled** waiter with ticket 0

1 a **preempted** waiter with ticket 1

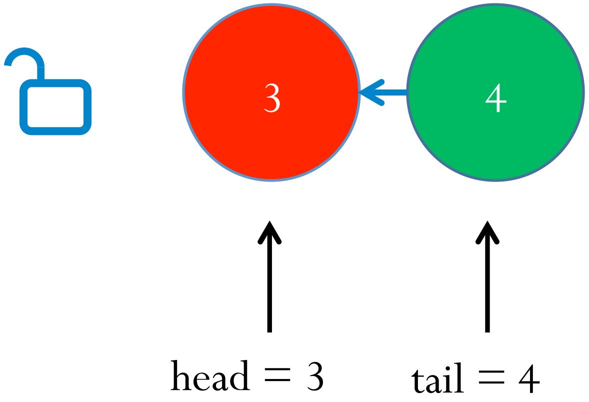
Preemption in Ticket Lock



🟢 0 a **scheduled** waiter with ticket 0

🔴 1 a **preempted** waiter with ticket 1

Preemption in Ticket Lock



3

4

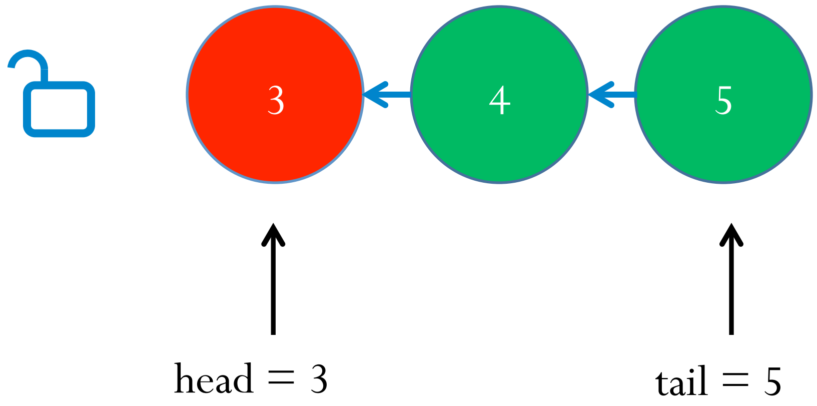
head = 3

tail = 4

🟢 0 a **scheduled** waiter with ticket 0

🔴 1 a **preempted** waiter with ticket 1

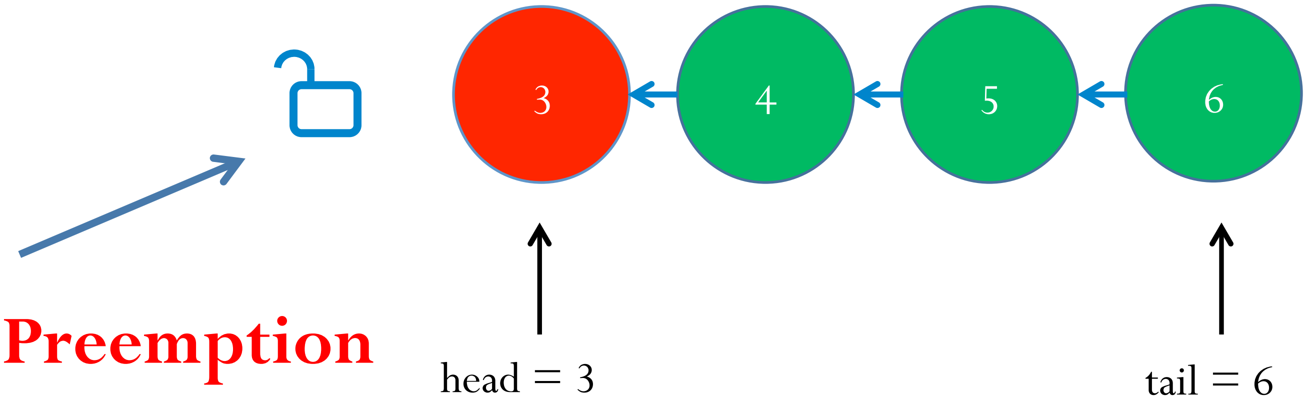
Preemption in Ticket Lock



🟢 0 a **scheduled** waiter with ticket 0

🔴 1 a **preempted** waiter with ticket 1

Preemption in Ticket Lock



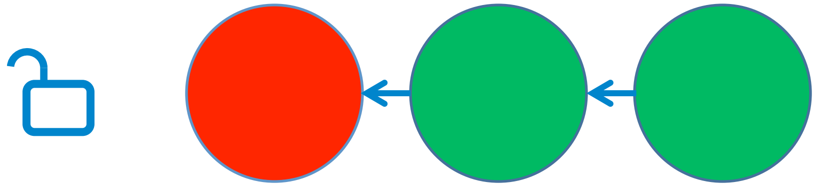
Lock Waiter Preemption
wait on available resource

🟢 0 a **scheduled** waiter with ticket 0

🔴 1 a **preempted** waiter with ticket 1

Lock Waiter Preemption

- Lock waiter is preempted
- Later waiters wait on an available lock
- Possible to adapt to it, if we
  - detect preempted waiter
  - acquire lock out of order



How significant is it??

Waiter Preemption Dominates

	LHP + LWP	**LWP**	**LWP/LHP +LWP**
hackbench x1	1089	452	41.5%
hackbench x2	44342	39221	**88.5%**
ebizzy x1	294	166	56.5%
ebizzy x2	1017	980	**96.4%**

Table 2: Lock Waiter Preemption Problem in the Linux Kernel

Lock *waiter* preemption dominates in overcommitted environments

Challenges & Approach

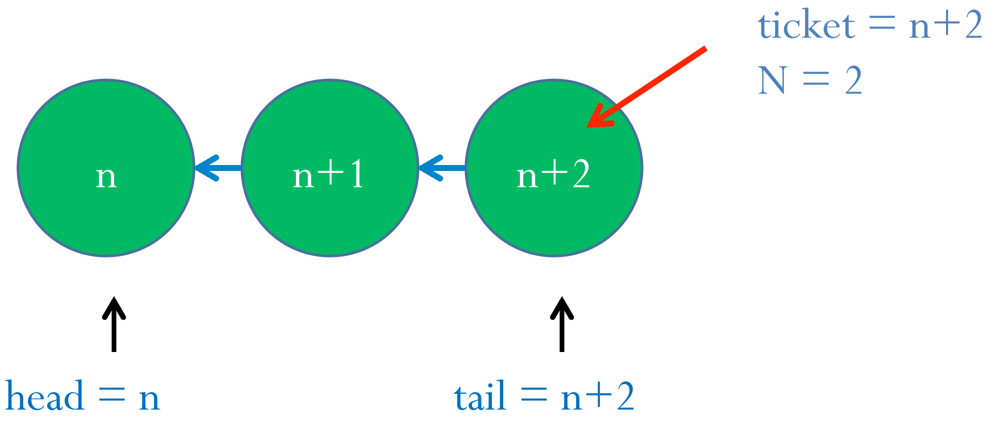
- How to identify a preempted waiter?
  - *timeout threshold*

- How to violate order constraints?
  - allow timed out waiters get the lock randomly
  - ensure mutual exclusion between them
- How NOT to break the whole ordering mechanism?
  - timeout threshold *proportional* to queue position

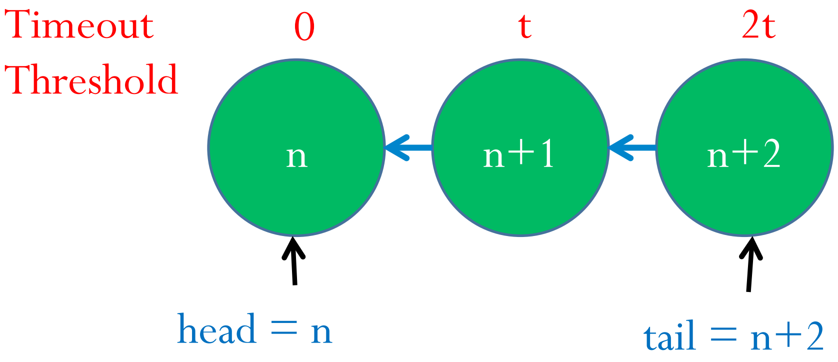
Queue Position Index

$$N = \text{ticket} - \text{queue\_head}$$

- ticket: copy of queue tail value upon enqueue
- N: number of earlier waiters



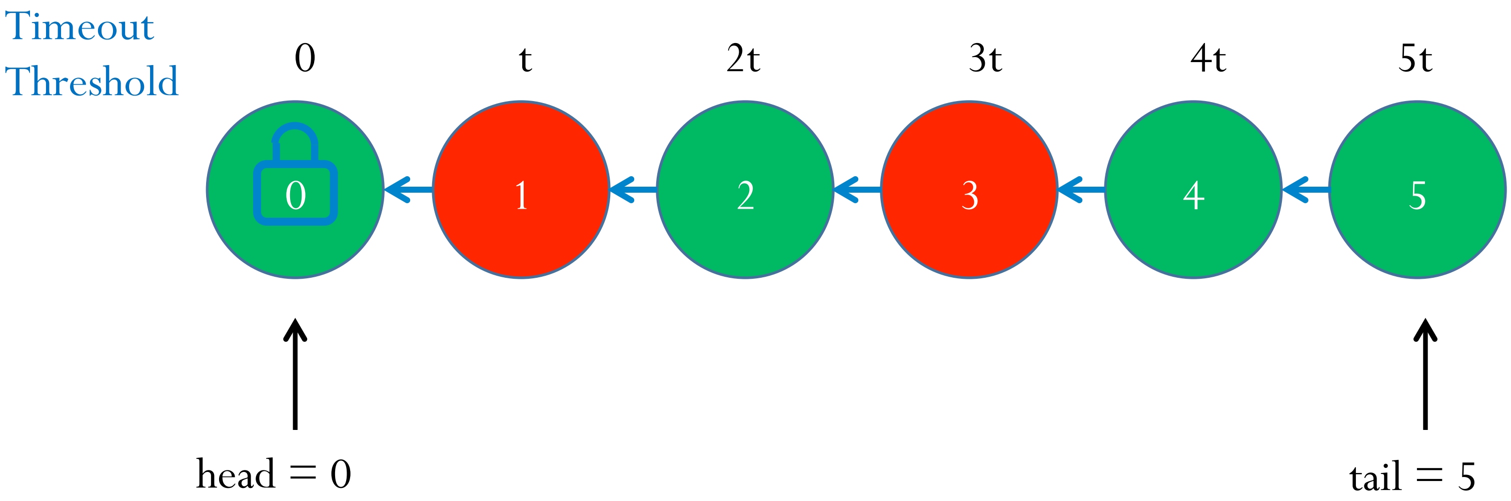
Proportional Timeout Threshold



$T = N \times t$

- t is a constant parameter
  - large enough to avoid false detection
  - small enough to save waiting time
- Performance is NOT t value sensitive
  - most locks take ~1us & most spinning time wasted on locks that wait ~1ms
    - larger t does not harm & smaller t does not gain much

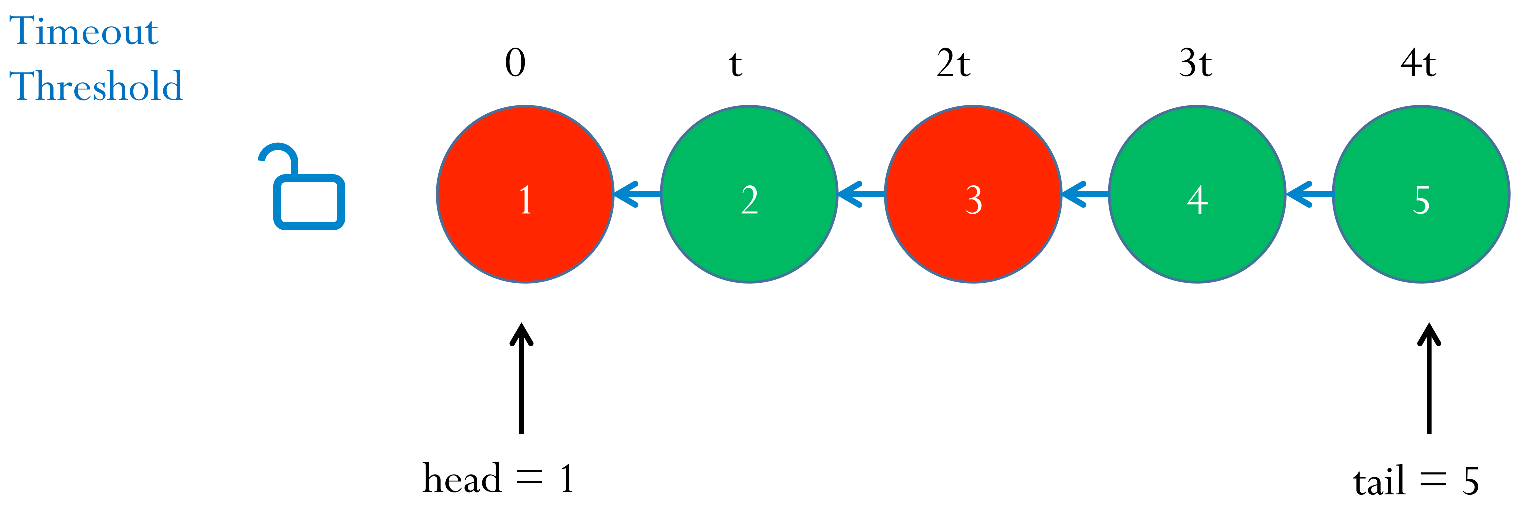
Preemptable Ticket Spinlock



🟢 0 a **scheduled** waiter with ticket 0

🔴 1 a **preempted** waiter with ticket 1

Preemptable Ticket Spinlock

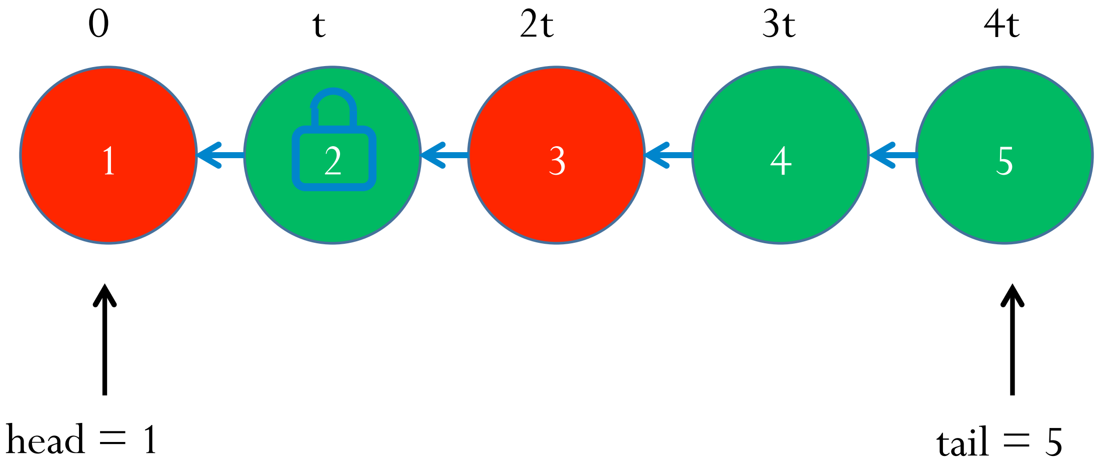


0 a **scheduled** waiter with ticket 0

1 a **preempted** waiter with ticket 1

Preemptable Ticket Spinlock

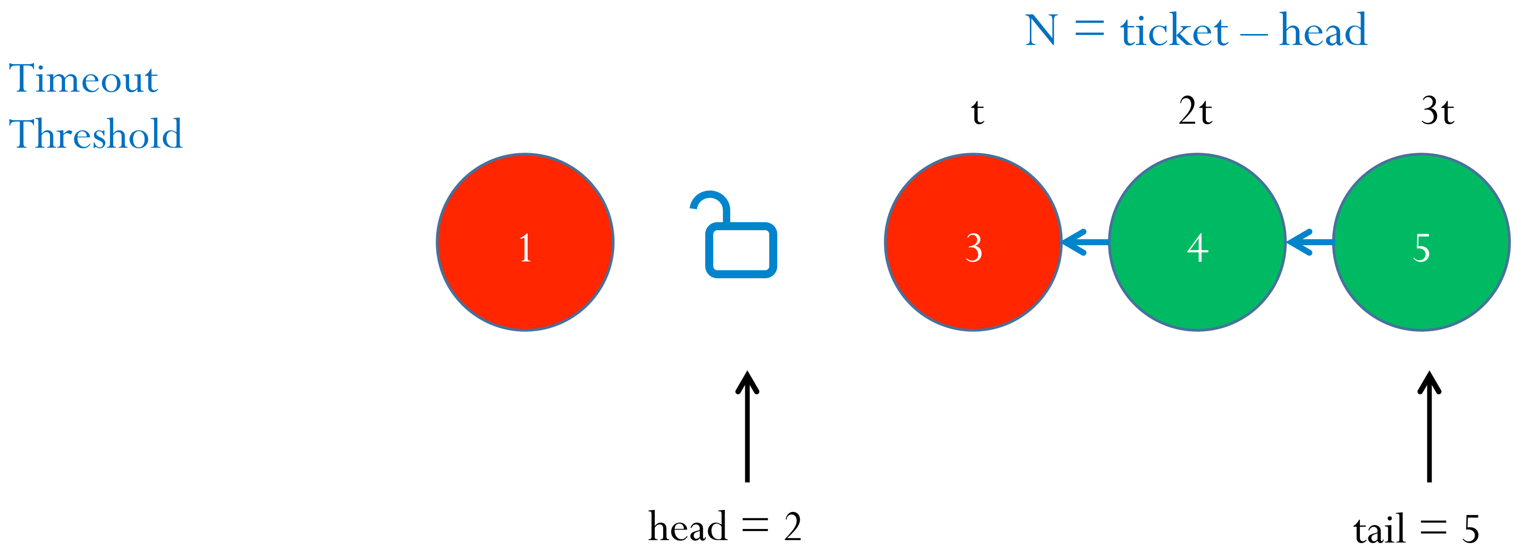
Timeout
Threshold



0 a **scheduled** waiter with ticket 0

1 a **preempted** waiter with ticket 1

Preemptable Ticket Spinlock



N = ticket – head

Timeout
Threshold

t

2t

3t

1

3

4

5

head = 2

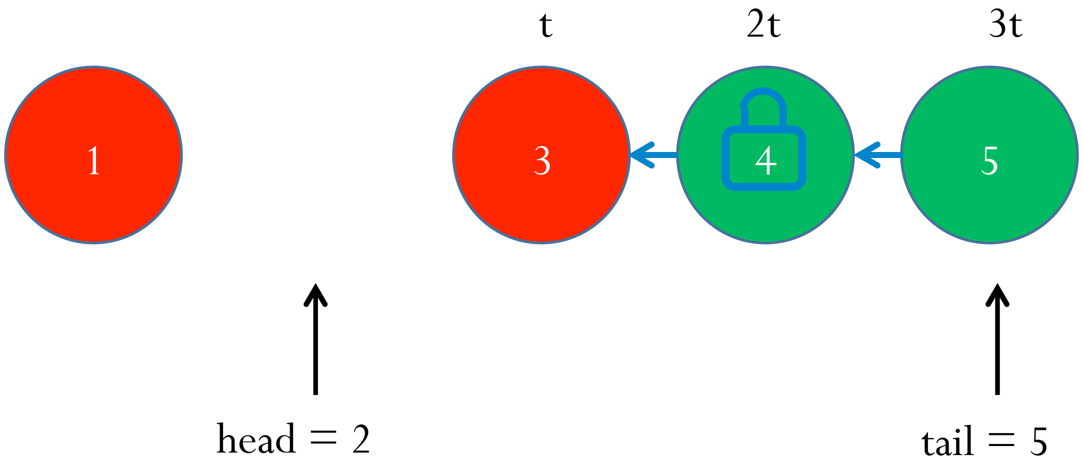
tail = 5

🟩 0 a **scheduled** waiter with ticket 0

🟥 1 a **preempted** waiter with ticket 1

Preemptable Ticket Spinlock

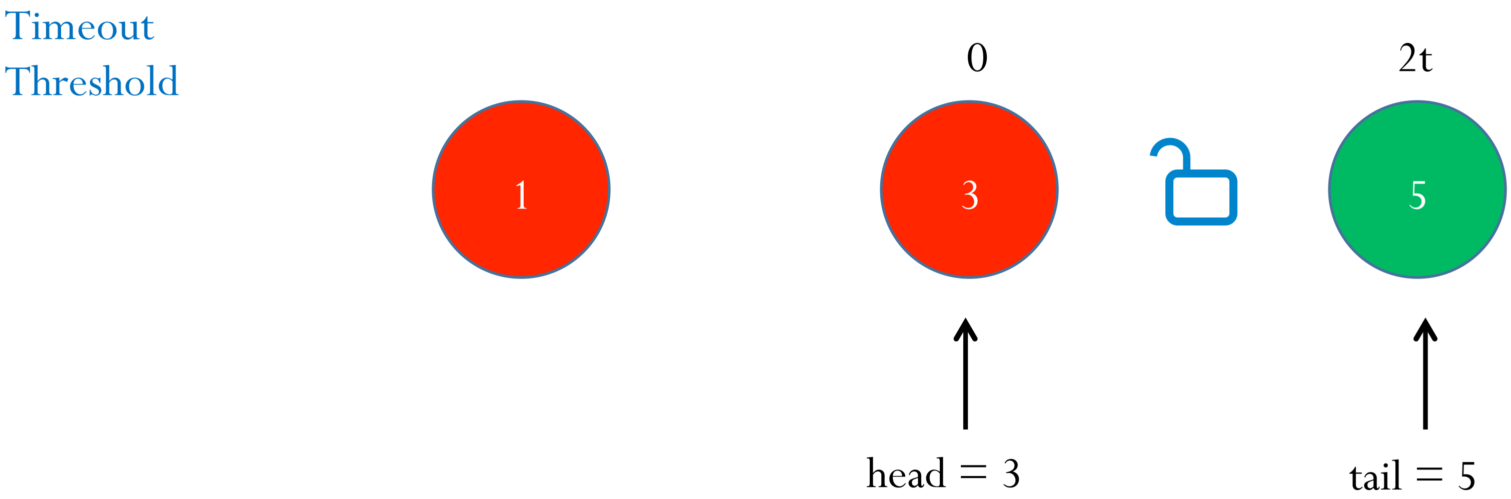
Timeout
Threshold



🟢 0 a **scheduled** waiter with ticket 0

🔴 1 a **preempted** waiter with ticket 1

Preemptable Ticket Spinlock



Timeout
Threshold

0

2t

1

3

5

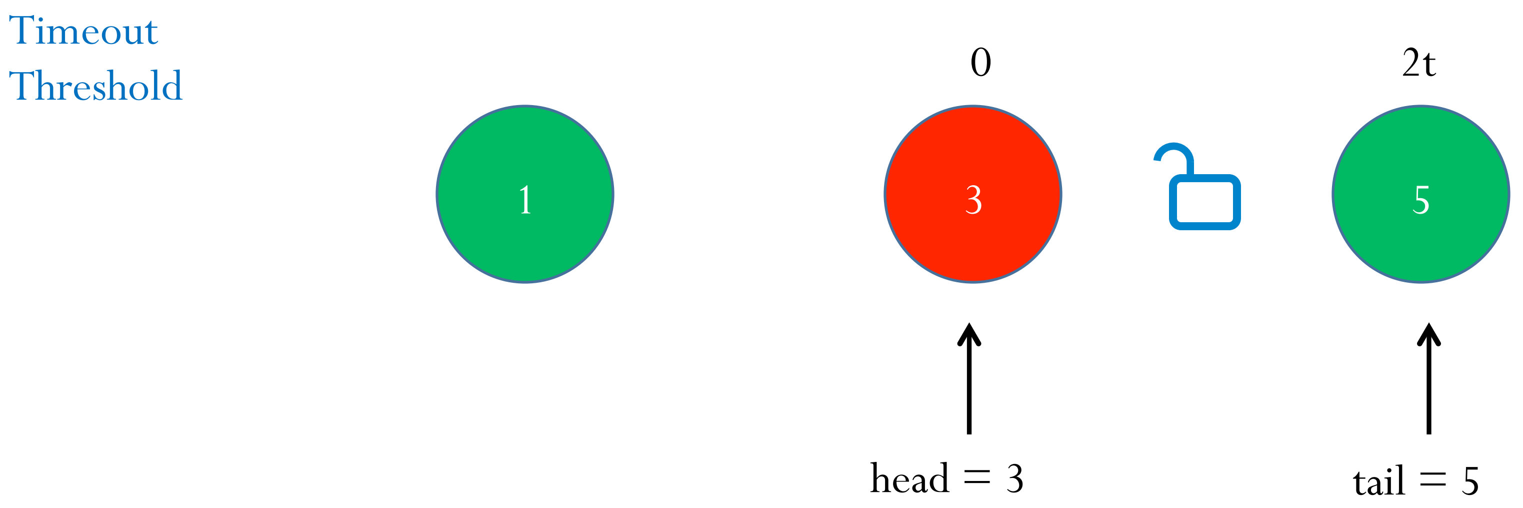
head = 3

tail = 5

🟩 0 a **scheduled** waiter with ticket 0

🟥 1 a **preempted** waiter with ticket 1

Preemptable Ticket Spinlock

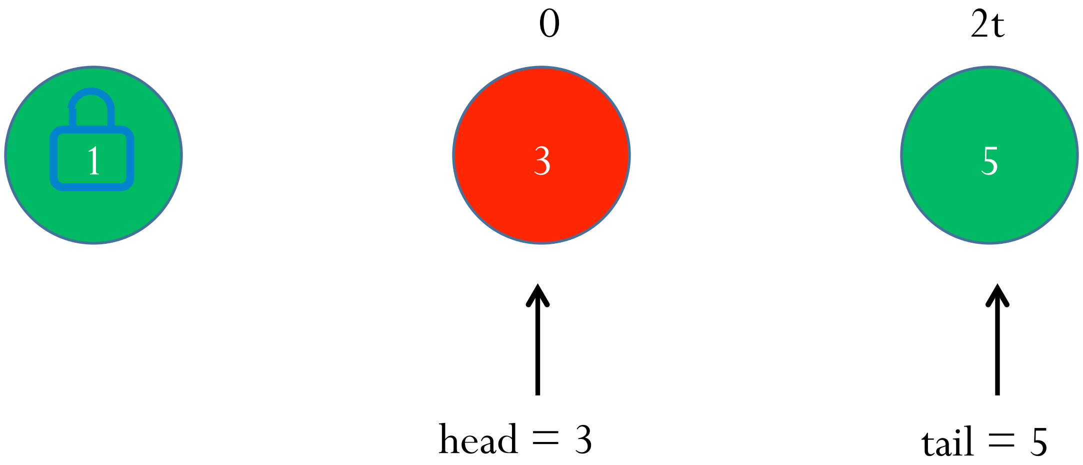


🟩 0 a **scheduled** waiter with ticket 0

🟥 1 a **preempted** waiter with ticket 1

Preemptable Ticket Spinlock

Timeout
Threshold



🟢 0 a **scheduled** waiter with ticket 0

🔴 1 a **preempted** waiter with ticket 1

Summary

- Preemptable Ticket Lock adapts to preemption
  - preserve order in absence of preemption
  - violate order upon preemption

- Preemptable Ticket Lock preserves fairness
  - order violations are restricted
    - priority is always given to timed out waiters
    - timed out waiters bounded by vCPU numbers of a VM

Implementation

- Drop-in replacement
  - lock(), unlock(), is_locked(), trylock(), etc.
- Correct
  - race condition free: atomic updates
- Fast
  - performance is sensitive to lock efficiency
- ~60 lines of C/inline-assembly in Linux 3.5.0

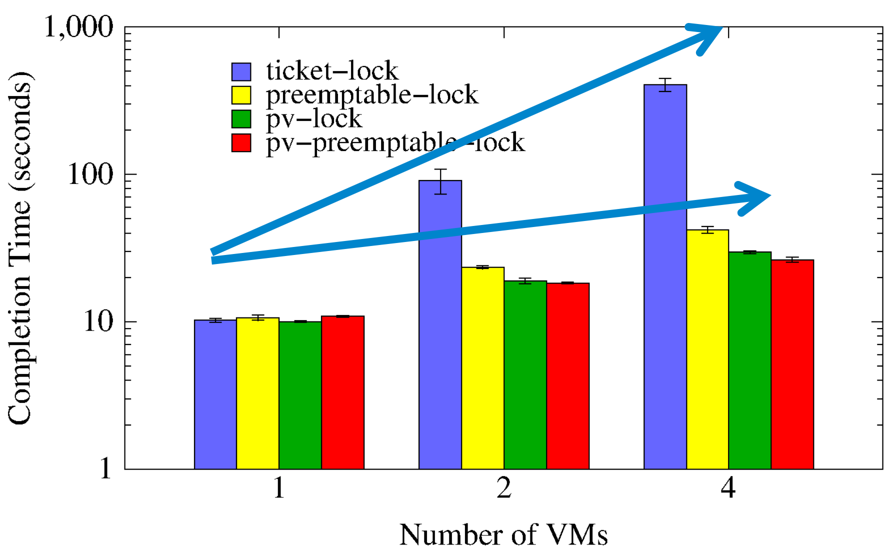
Paravirtual Spinlocks

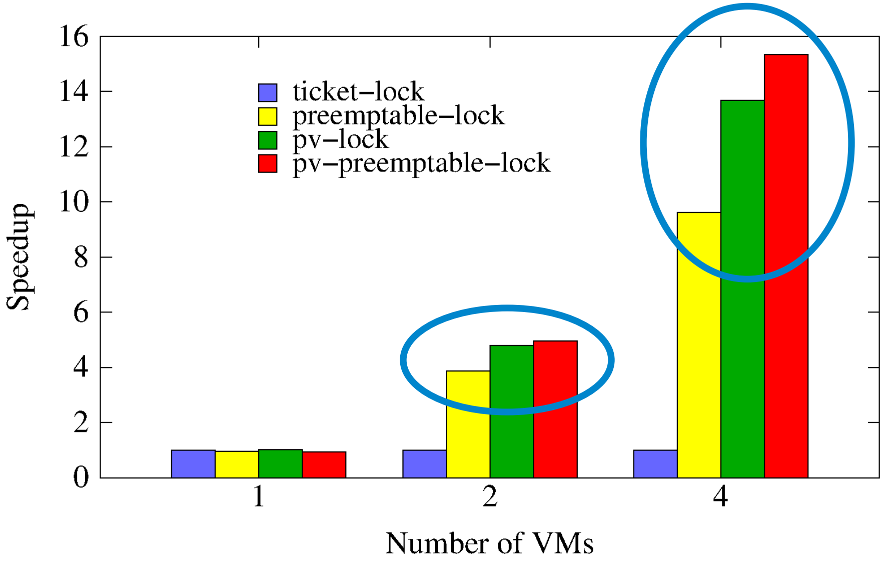
- Lock holder preemption is unaddressed
  - semantic gap between guest and host
  - paravirtualization: guest/host cooperation
    - signal long waiting lock / put a vCPU to sleep
    - notify to wake up a vCPU / wake up a vCPU
  - paravirtual preemptable ticket spinlock
    - sleep when waiting too long after timed out
    - wake up all sleeping waiters upon lock releasing

Evaluation

- Host
  - 8 core 2.6GHz Intel Core i7 CPU, 8 GB RAM, 1Gbit NIC, Fedora 17 (Linux 3.5.0)
- Guest
  - 8 core, 1G RAM, Fedora 17 (Linux 3.5.0)
- Benchmarks
  - hackbench, ebizzy, dell dvd store
- Lock implementations
  - baseline: ticket lock, paravirtual ticket lock (pv-lock)
  - preemptable ticket lock
  - paravirtual (pv) preemptable ticket lock

Hackbench



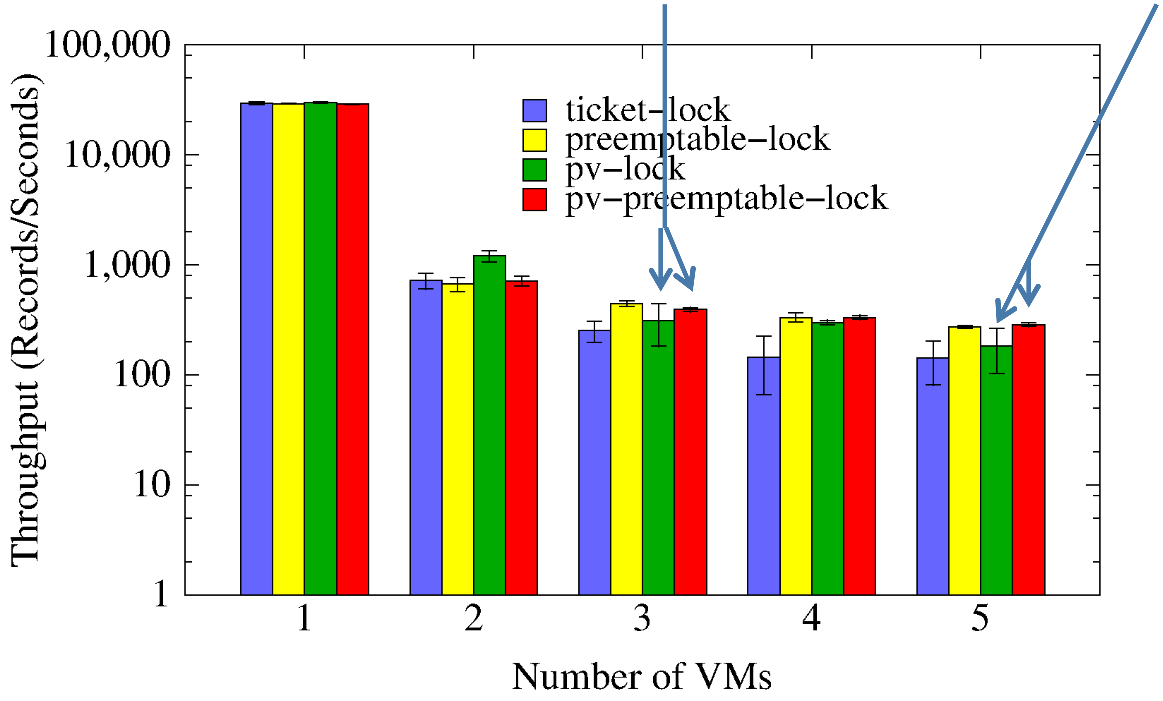


- Average Speedup
  - preemptable-lock vs. ticket lock: 4.82X
  - pv-preemptable-lock v.s. ticket lock: 7.08X
  - pv-preemptable-lock v.s. pv-lock: 1.03X

Ebizzy

variance
131.62 vs. 16.09

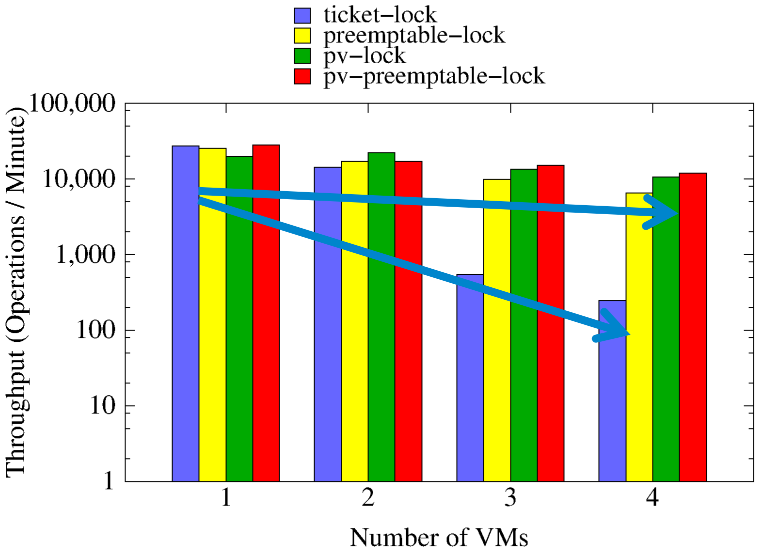
variance
80.36 vs. 10.94

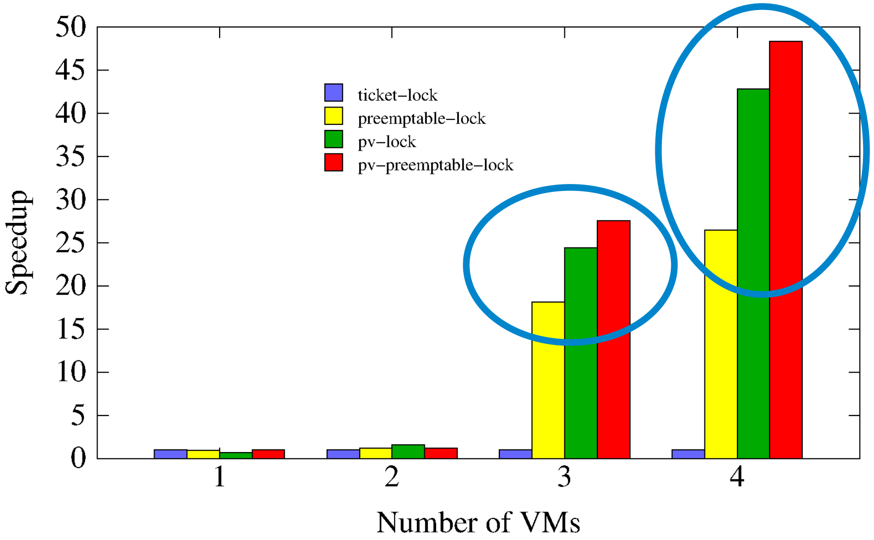


Less variance over ticket lock and pv-lock

- in-VM preemption adaptivity
- less VM interference

Dell DVD Store (apache/mysql)





- Average Speedup
  - preemptable-lock vs. ticket lock: 11.68X
  - pv-preemptable-lock v.s. ticket lock: 19.52X
  - pv-preemptable-lock v.s. pv-lock: 1.11X

Evaluation Summary

- Preemptable Ticket Spinlocks speedup
  - 5.32X over ticket lock

- Paravirtual Preemptable Ticket Spinlocks speedup
  - 7.91X over ticket lock
  - 1.08X over paravirtual ticket lock

Average speedup across cases for all benchmarks

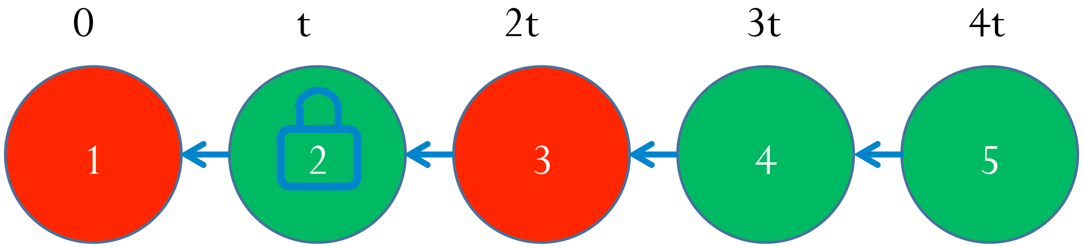
Conclusion

- Lock Waiter Preemption
  - most significant preemption problem in queue based lock under overcommitted environment
- Preemptable Ticket Spinlock
  - Implementation with ~60 lines of code in Linux
- Better performance in overcommitted environment
  - 5.32X average speedup up over ticket lock w/o VMM support
  - 1.08X average speedup over pv-lock with less variance

Thank You

- **Jiannan Ouyang**
  - ouyang@cs.pitt.edu
  - http://www.cs.pitt.edu/~ouyang/

Preemptable Ticket Spinlock


0
t
2t
3t
4t
1
2
3
4
5