AMD
Smarter Choice

How to Deal with Lock-Holder Preemption

Thomas Friebel

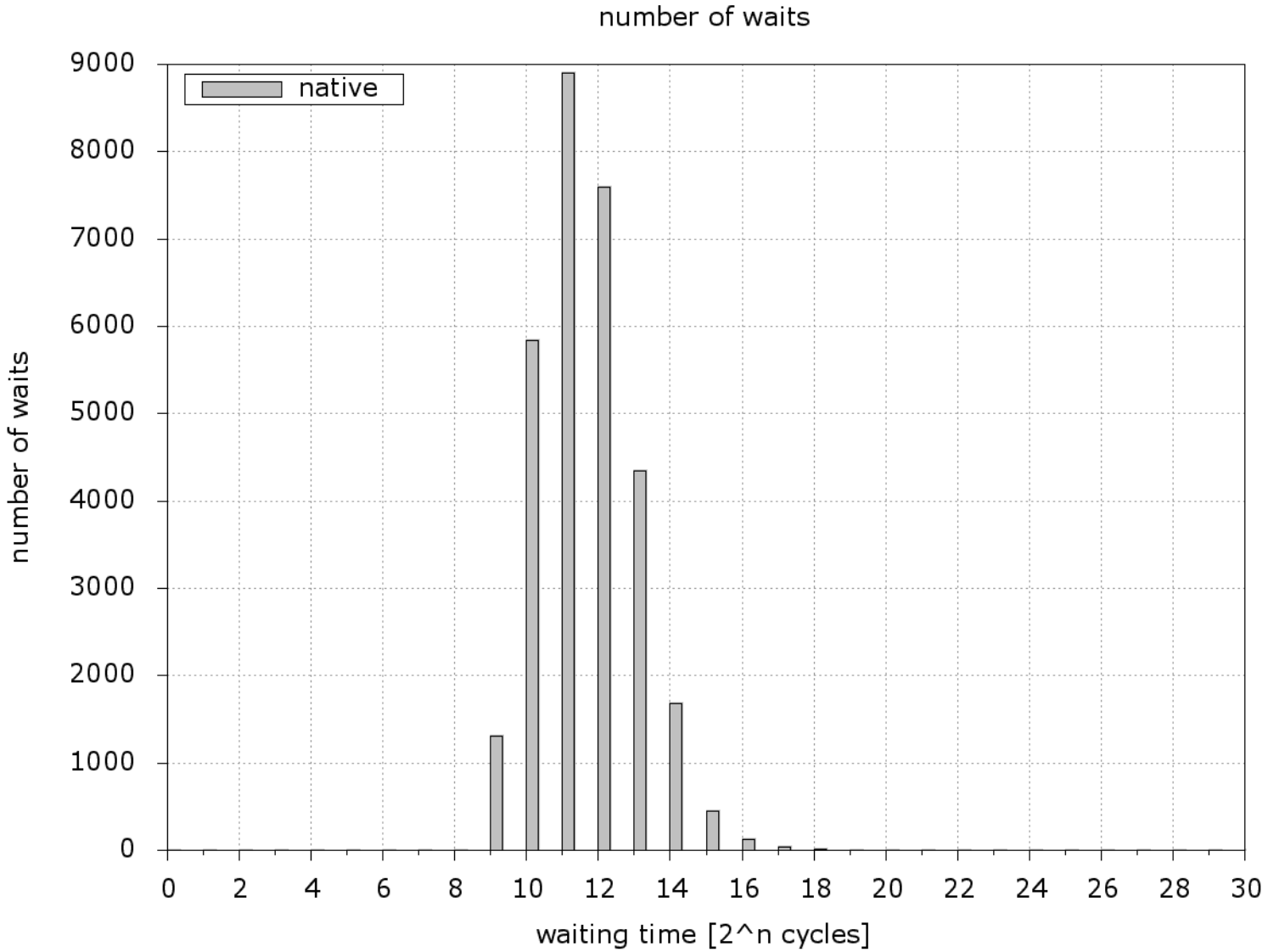
October 2008

Spinlock Basics

Spinlocks wait actively as opposed to sleeping locks

Used for short critical sections

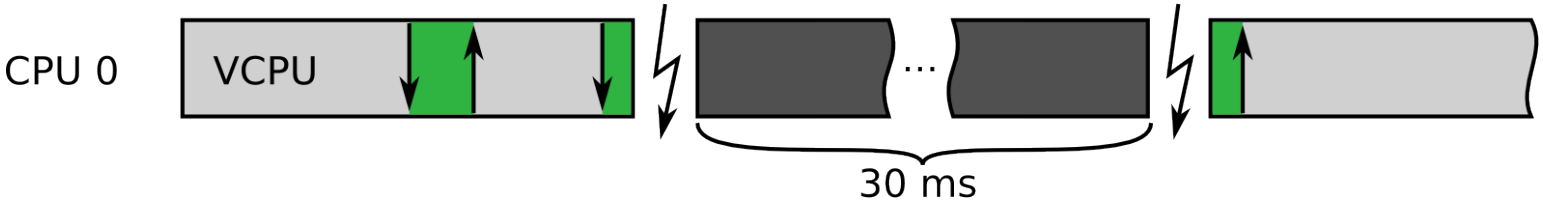
Spinlock Wait Times – Kernbench



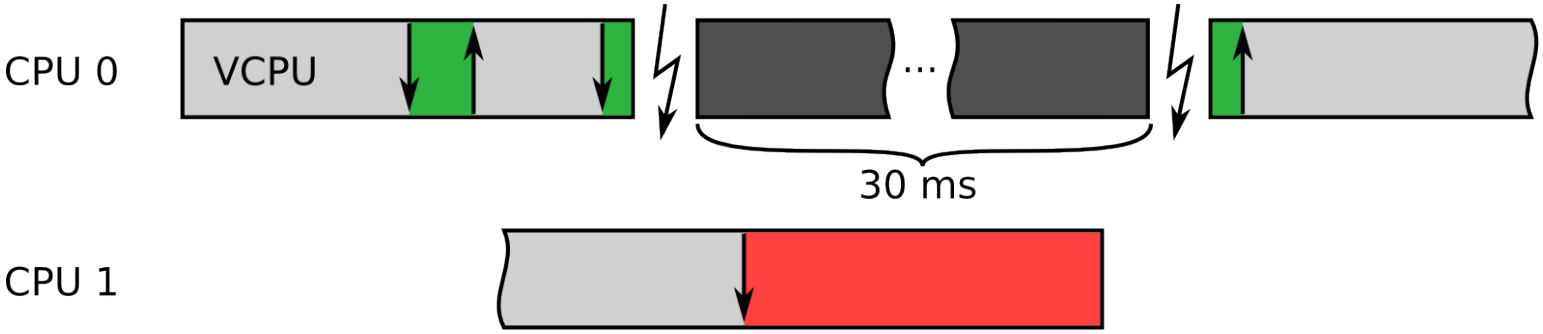
Spinlocks and Virtualization

Spinlocks and Virtualization

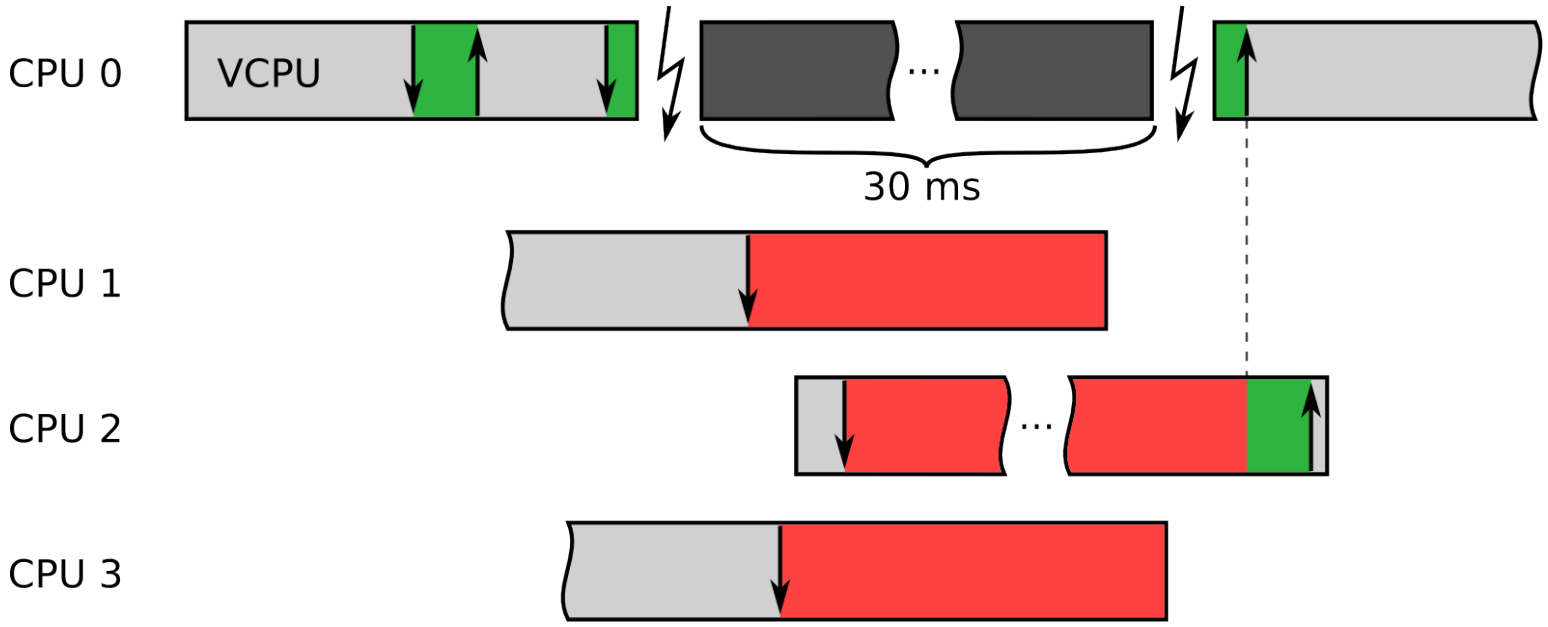
Spinlocks and Virtualization



Spinlocks and Virtualization



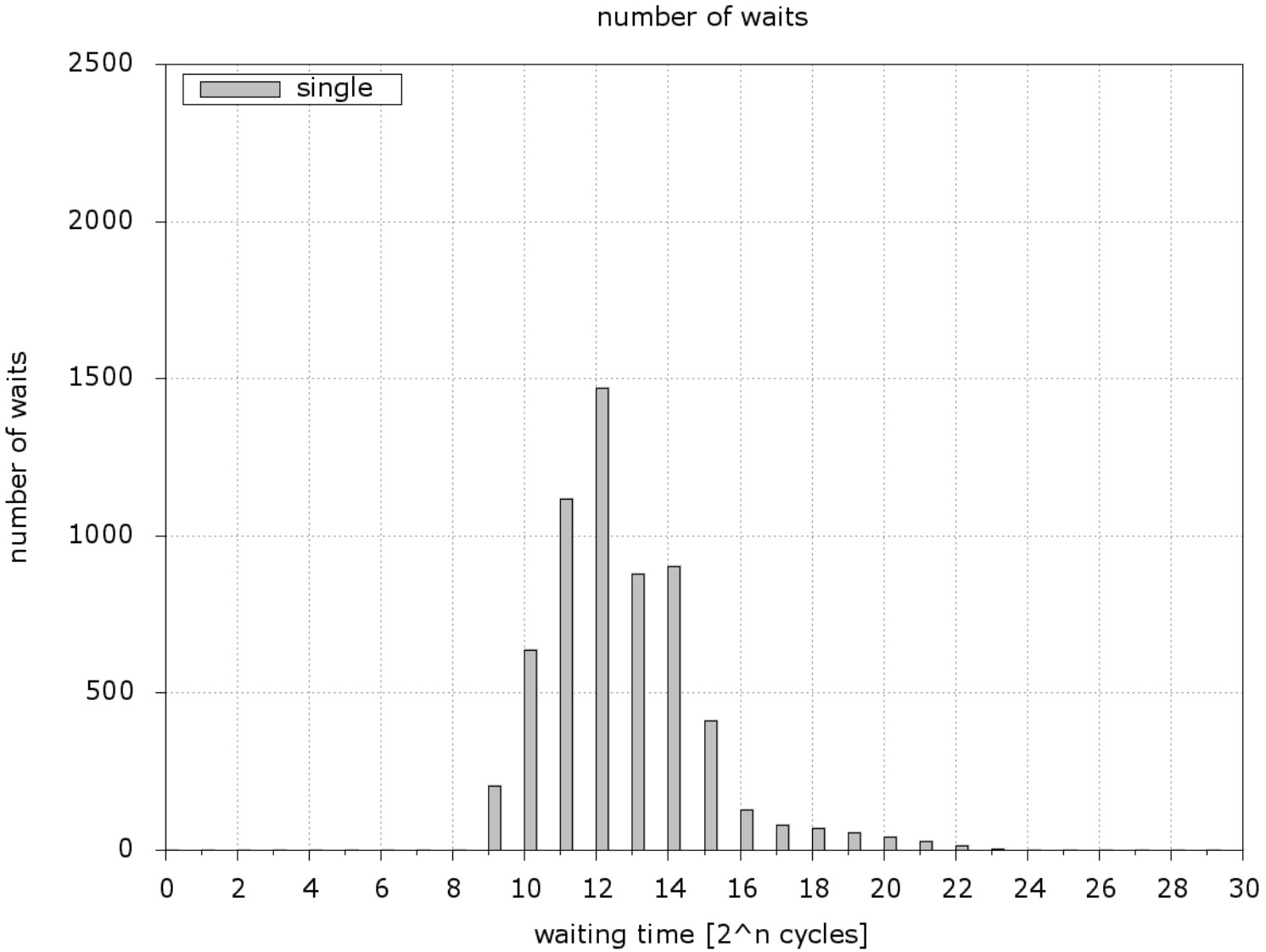
Spinlocks and Virtualization



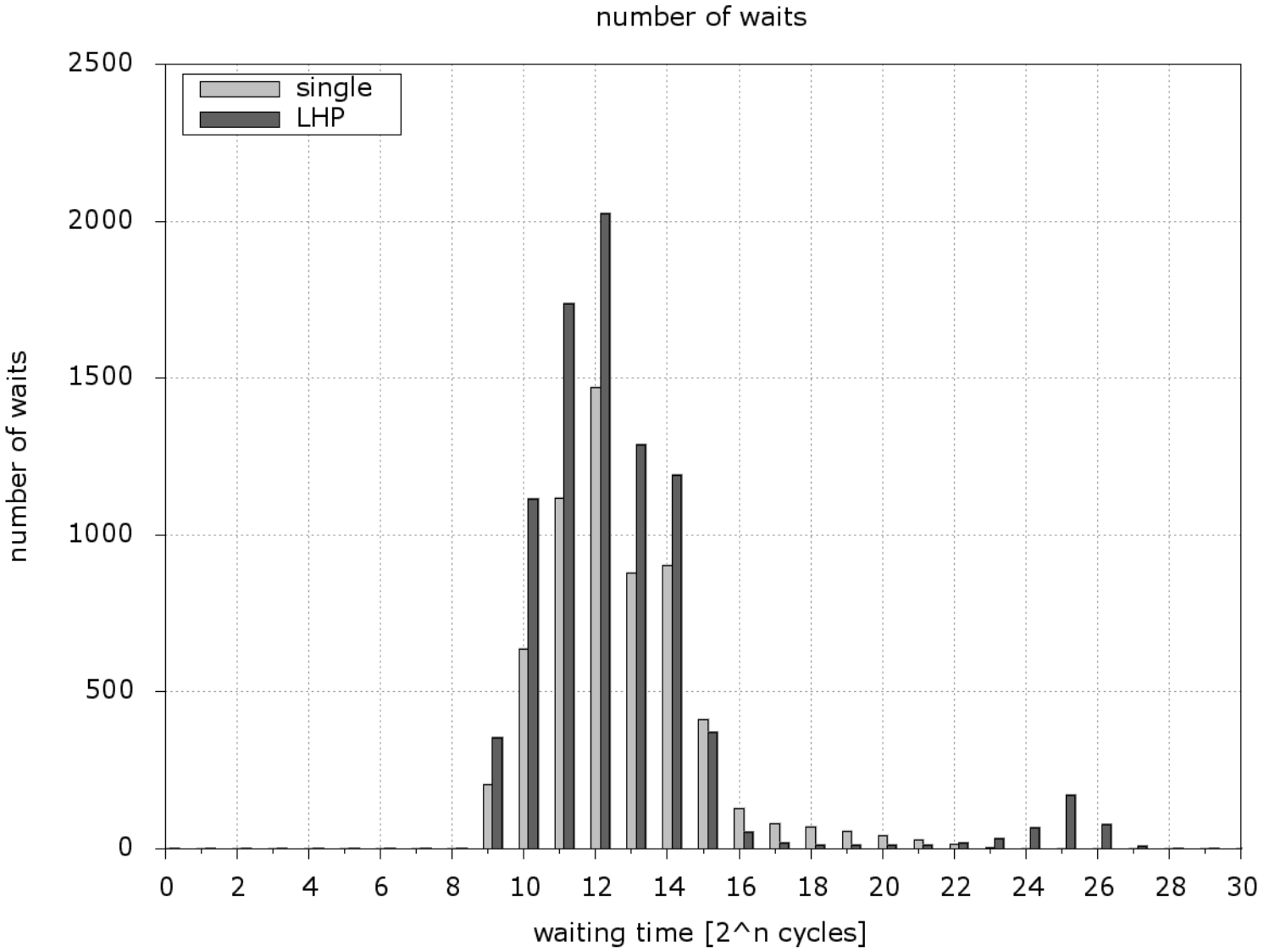
AMD
Smarter Choice

Is lock-holder preemption problematic?

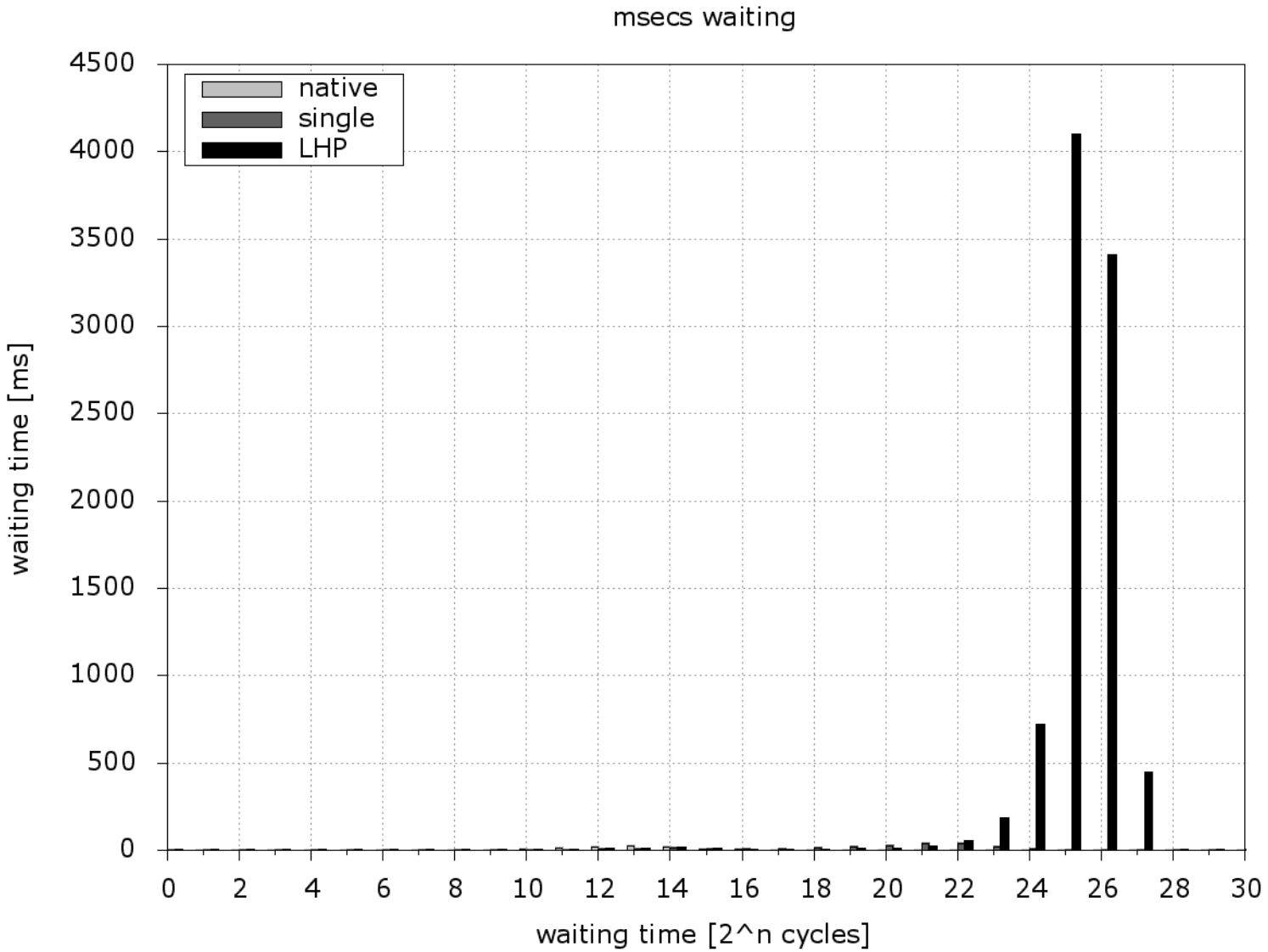
Kernbench in a Guest



Kernbench vs. 'while(true)'



Time, not Times



And in Numbers?

AMD Smarter Choice

	guest time [s]	time spent spinning [s]	[%]
single kernbench	109.0	0.2	0.2%
kernbench vs while(1)	117.3	9.0	7.6%
difference	7.6%		

AMD
Smarter Choice

What can we do about it?

Dealing with lock-holder preemption

AMD Smarter Choice

LHP avoidance

- No spinlock held in userspace
- Idea: Avoid preempting guest in kernel space
- Postpone guest switch to kernel exit
- Problem: extraordinary long critical sections, e.g. Apache using sendfile()

Helping locks

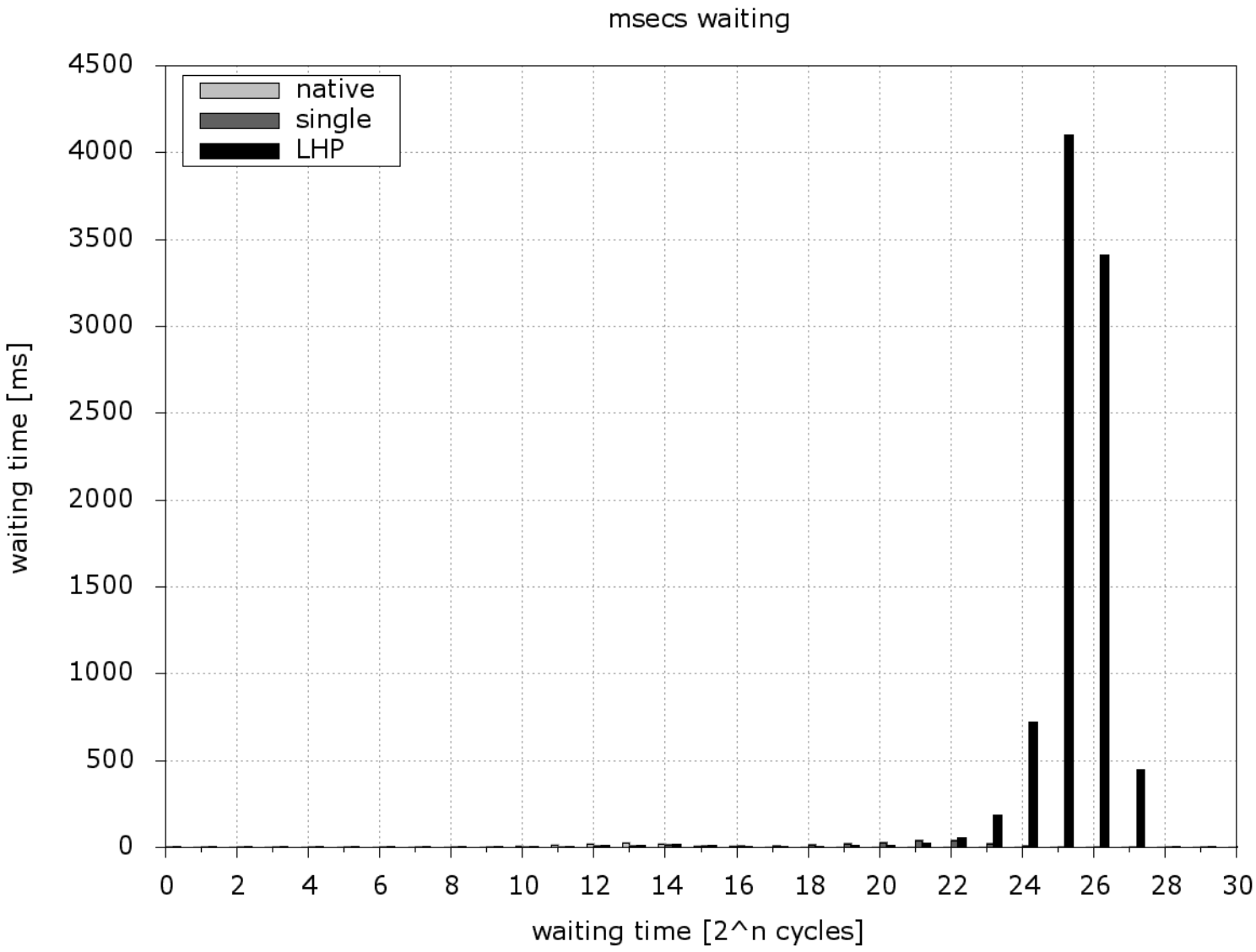
- Instead of busy waiting, switch to preempted lock-holder
- Problem: finding the preempted lock-holder

Helping locks: Ingredients

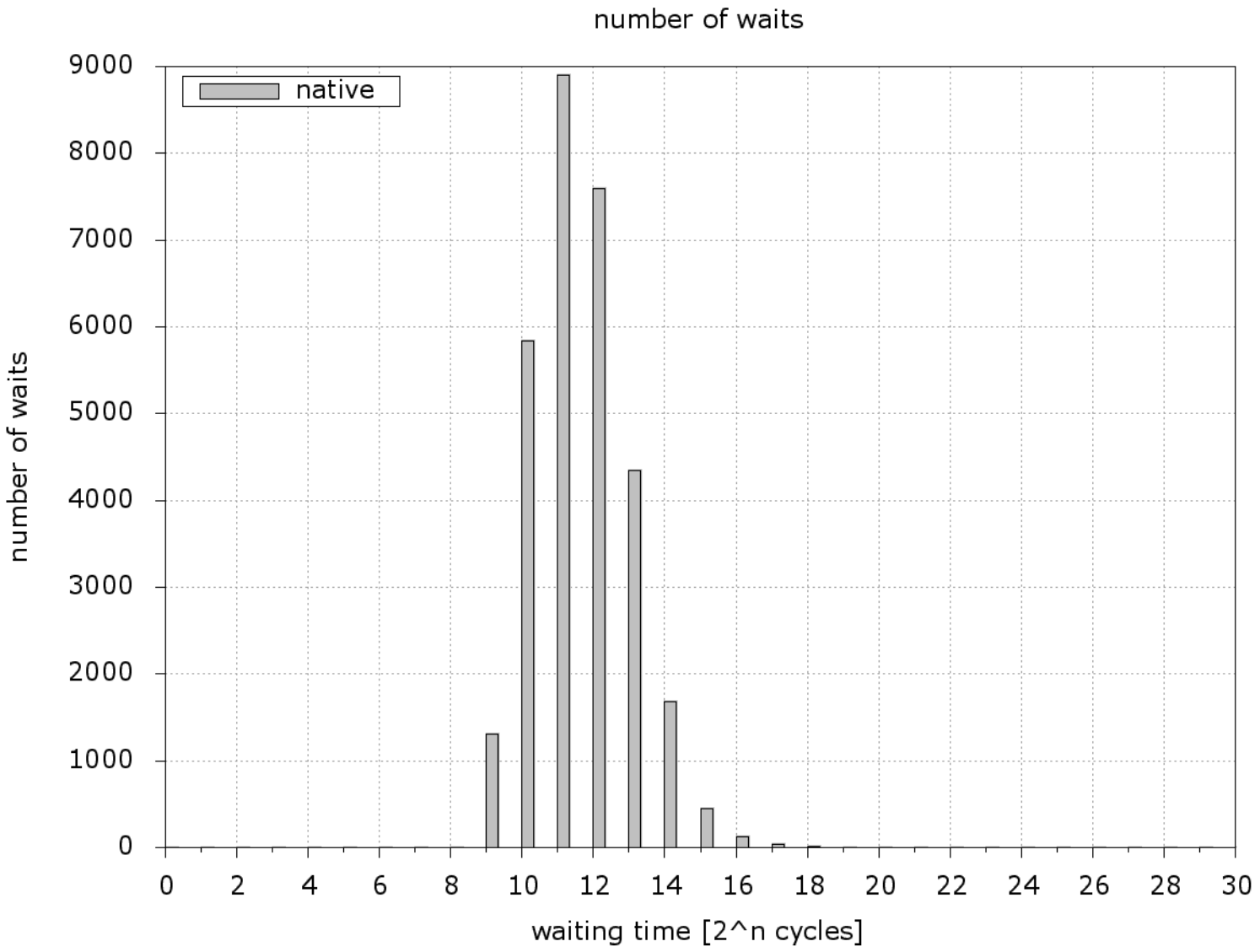
AMD Smarter Choice

1) Guest kernel: new 'yield' hypercall when waiting unusually long
   - Modify spinlock loop
2) Reasonable threshold for 'unusually long'
   - Histograms help
3) Selecting which VCPU to switch to

Threshold: Upper boundary



Threshold: Lower boundary



Scheduling Strategy

AMD Smarter Choice

Good choices:

- VCPUs of the same VM to make progress locally
- (Potential) preempted lock-holders
- Cache-„near“ VCPUs

Neither/nor:

- VCPUs in user space

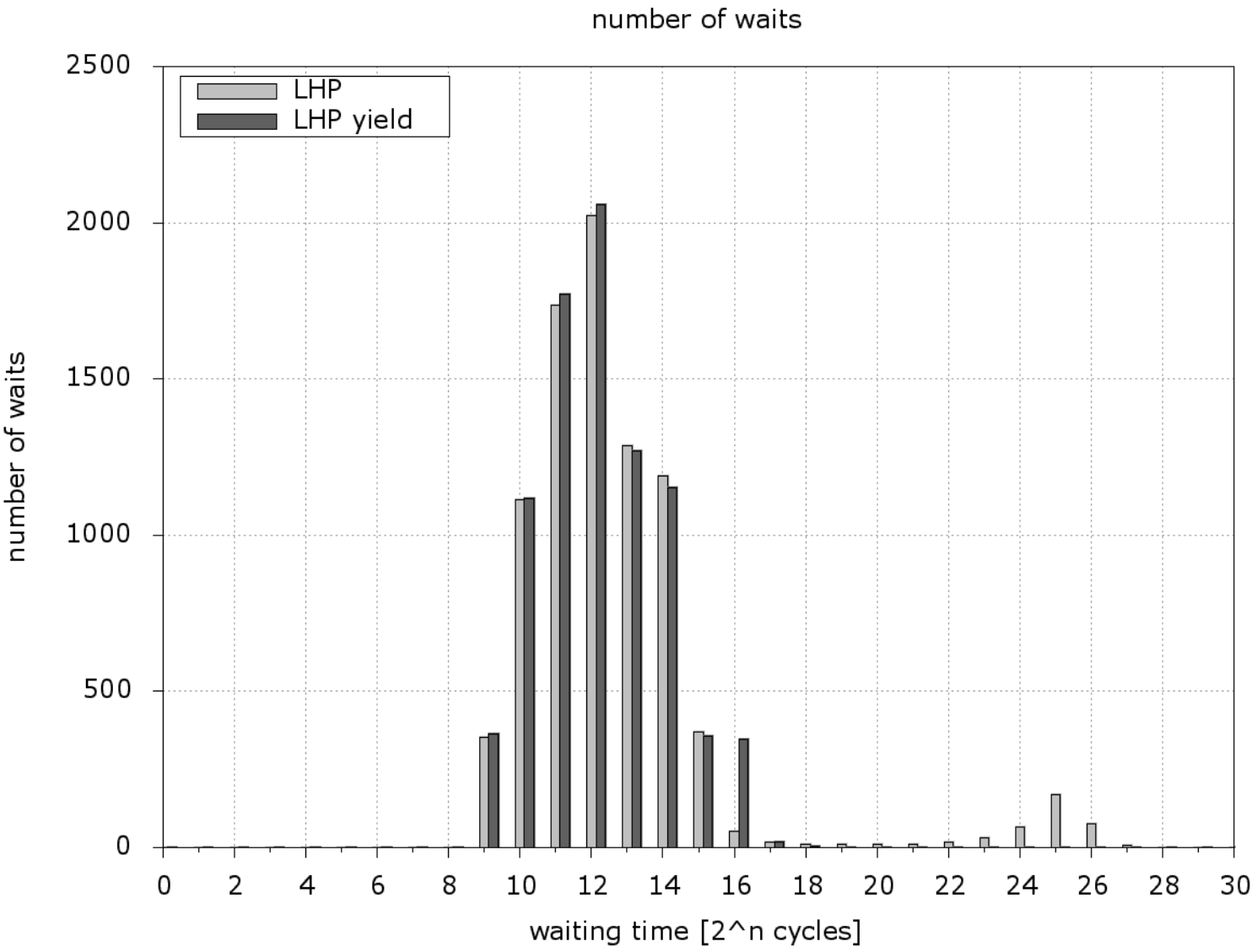
Bad choices:

- VCPUs which yielded recently

AMD
Smarter Choice

What about performance?

Histogram with 'yield' hypercall

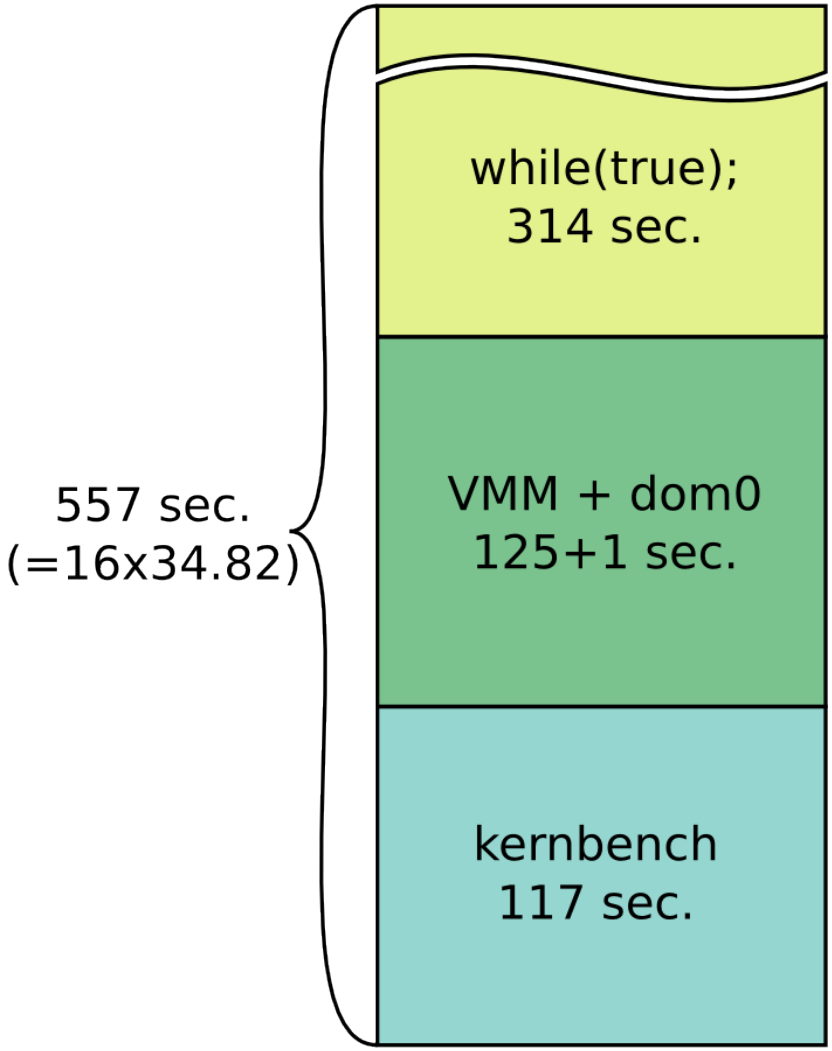


Performance

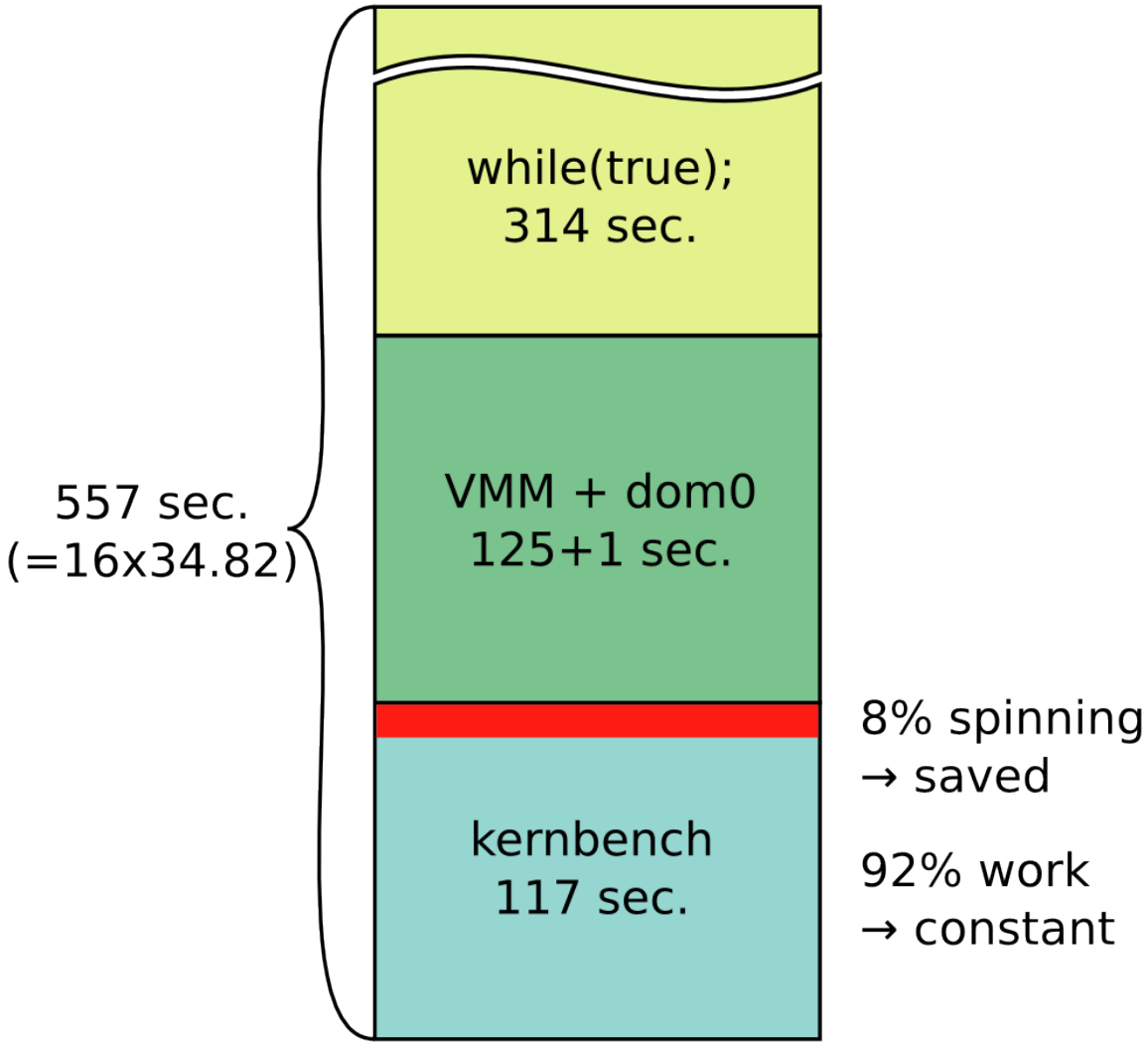
AMD Smarter Choice

	wall clock [s]	guest time [s]	time spent spinning [s]	[%]
LHP	34.8	117.3	9.0	7.6%
yield	33.5	108.4	0.0	0.0%
difference	-3.9%	-7.6%		-7.6%

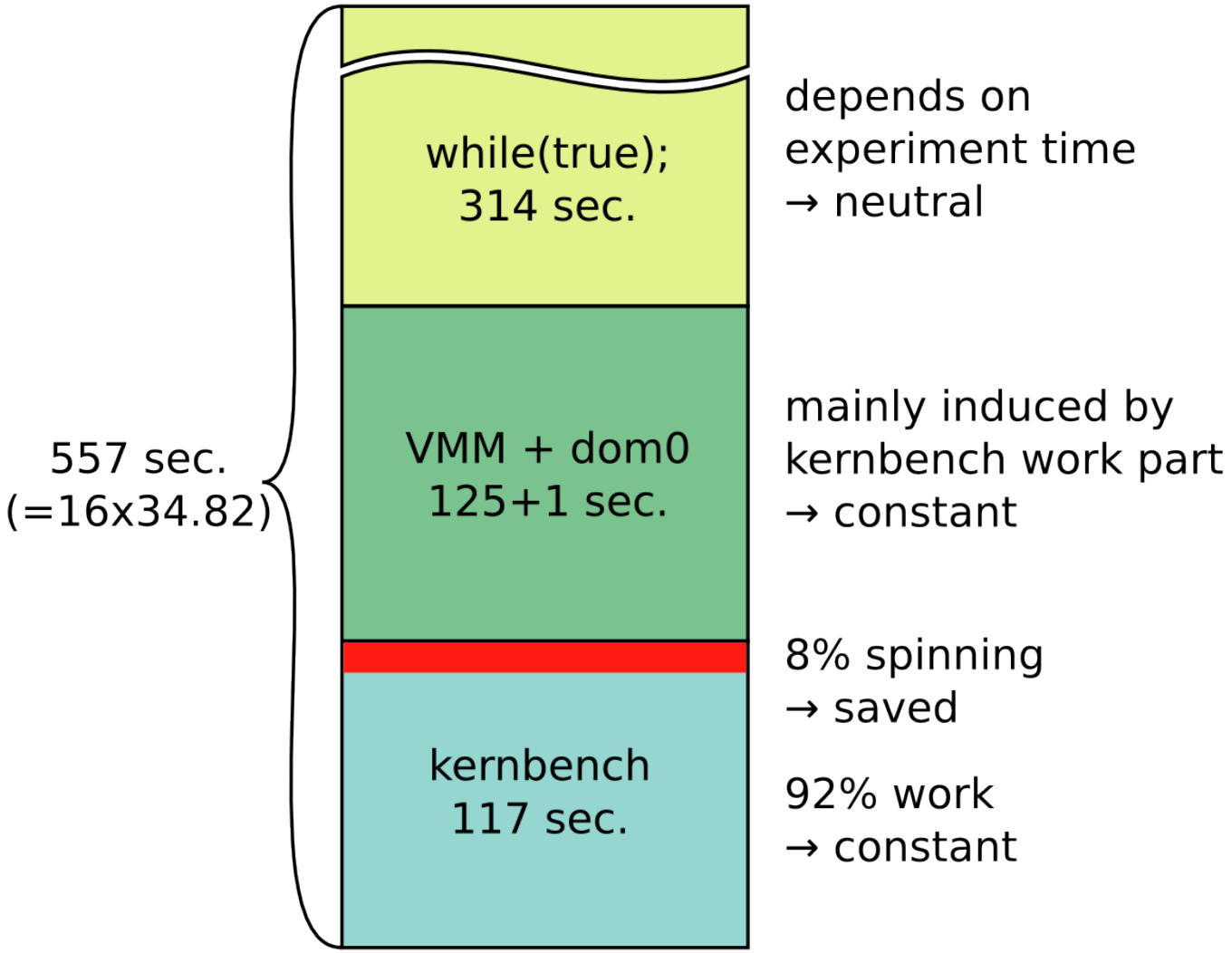
Efficiency


while(true);
314 sec.
VMM + dom0
125+1 sec.
kernbench
117 sec.
557 sec.
(=16x34.82)

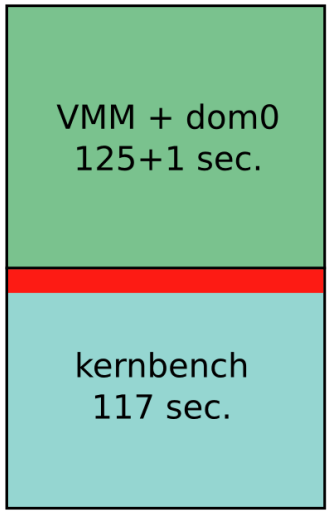
Efficiency


while(true);
314 sec.
VMM + dom0
125+1 sec.
kernbench
117 sec.
557 sec.
(=16x34.82)
8% spinning
→ saved
92% work
→ constant

Efficiency


while(true);
314 sec.
VMM + dom0
125+1 sec.
kernbench
117 sec.
557 sec.
(=16x34.82)
depends on
experiment time
→ neutral
mainly induced by
kernbench work part
→ constant
8% spinning
→ saved
92% work
→ constant

Efficiency



mainly induced by
kernbench work part
→ constant

8% spinning
→ saved

$$\frac{117\,sec}{117\,sec + 126\,sec} \times 7.6\% \;=\; 3.7\%$$

➔ Real result of 3.9% is reasonable

➔ Highly efficient

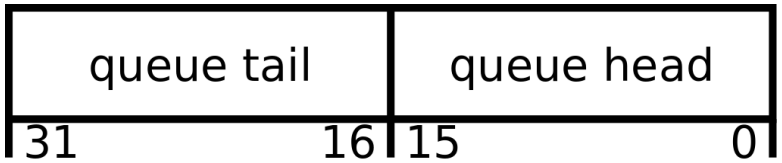
AMD
Smarter Choice

FIFO ticket spinlocks

FIFO ticket spinlocks

Next ticket in dispenser: queue tail

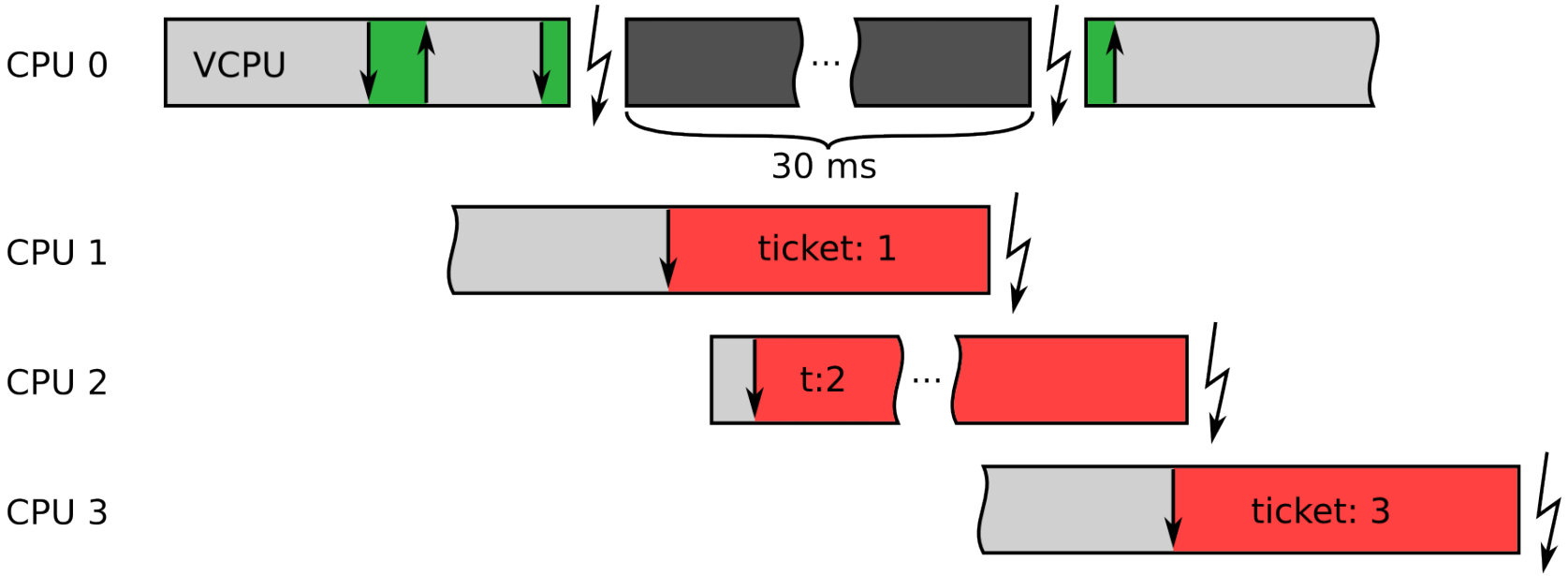
„Now serving“ display at counter: queue head



Lock: atomic(ticket = tail++); while (head != ticket);

Unlock: atomic(head++);

FIFO ticket spinlocks



Ticket locks and virtualization

	wall clock [s]	guest time [s]	time spent spinning [s]	[%]
LHP	2825.1	22434.2	22270.4	99.3%

Ticket locks and virtualization

AMD Smarter Choice

	wall clock [s]	guest time [s]	time spent spinning [s]	[%]
LHP	2825.1	22434.2	22270.4	99.3%
yield	34.1	123.6	6.6	5.4%

Conclusion

AMD Smarter Choice

Lock-holder preemption quite serious:
7.6% guest time wasted

Helping locks:
3.9% system performance improvement!
(Amdahl's law explains why)

New ticket spinlocks:
30 secs kernbench takes 45 minutes

Helping locks help here, too

AMD
Smarter Choice

Trademark Attribution