

A square graphic on the left side of the slide, divided diagonally from the top-left to the bottom-right. The top-right triangle is green, and the bottom-left triangle is black.

Preventing Guests from Spinning Around

How to Deal with Lock-Holder Preemption
Thomas Friebe

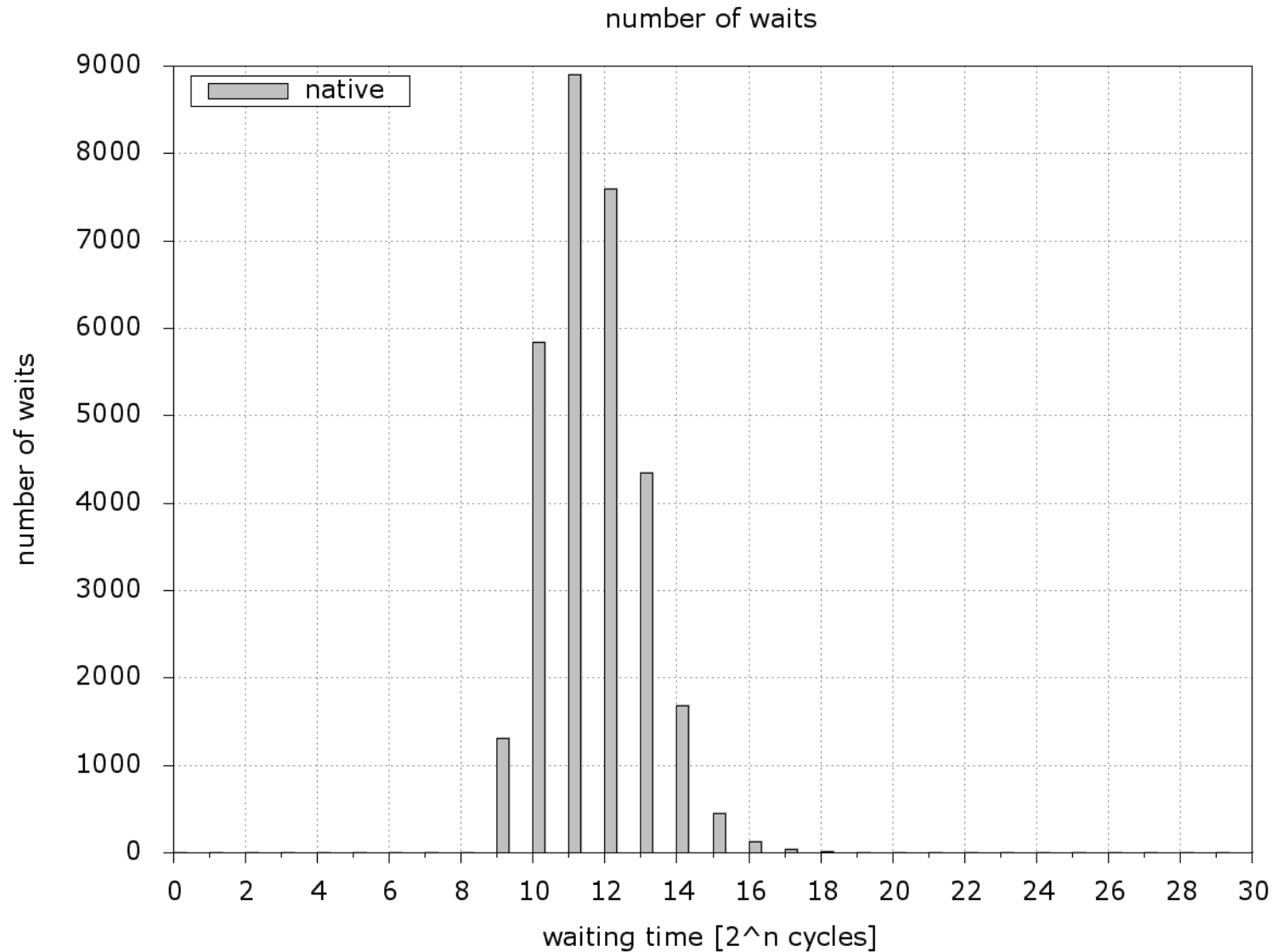
June 23, 2008

Spinlock Basics

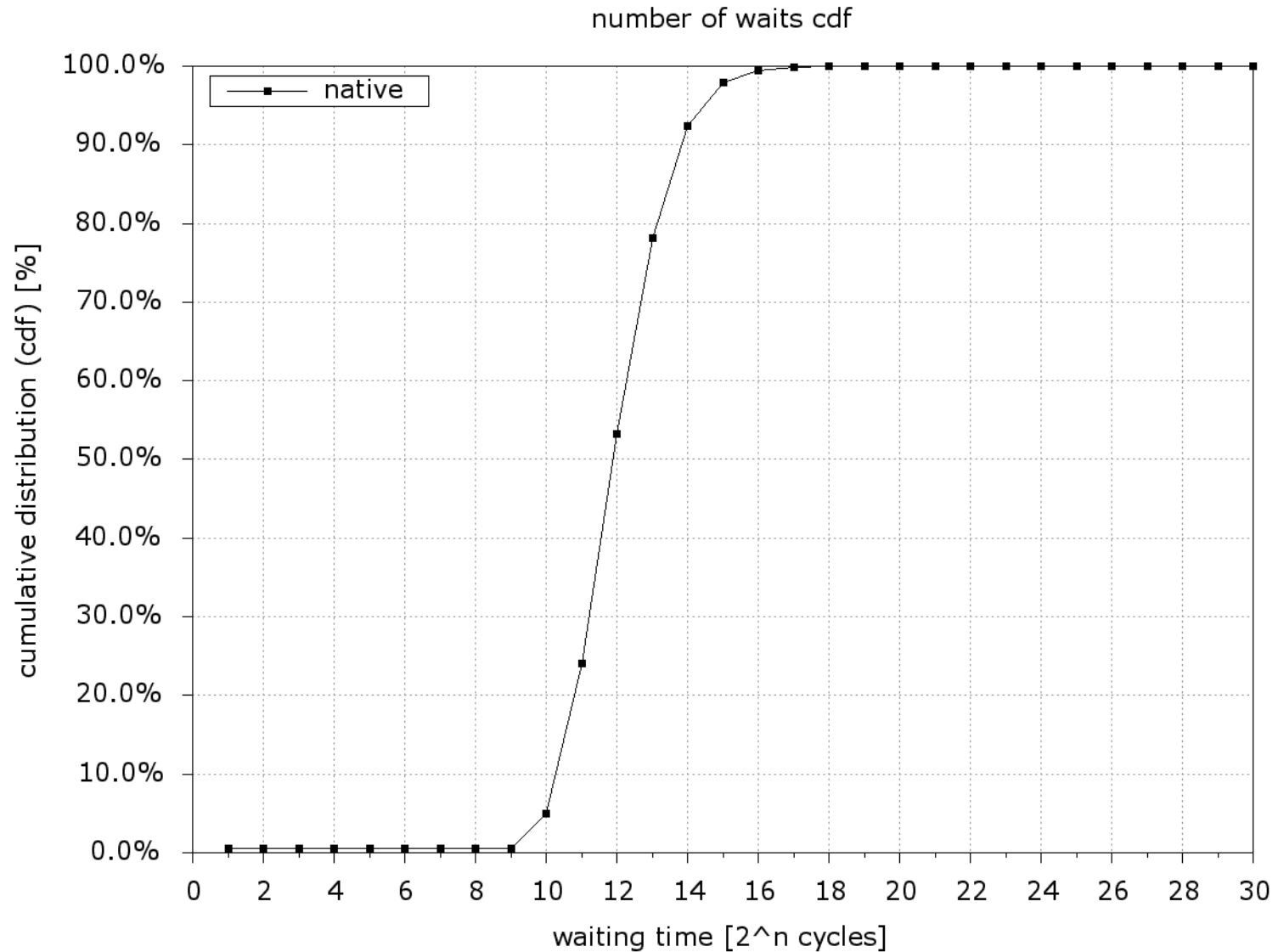
Spinlocks wait actively as opposed to sleeping locks

Used for short critical sections

Spinlock Wait Times – Kernbench



Spinlock Wait Times – Kernbench

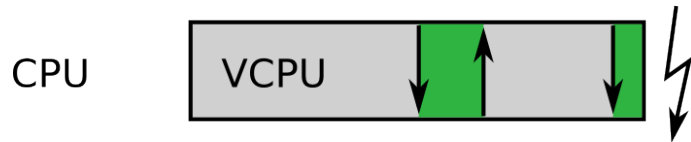


Spinlocks and Virtualization

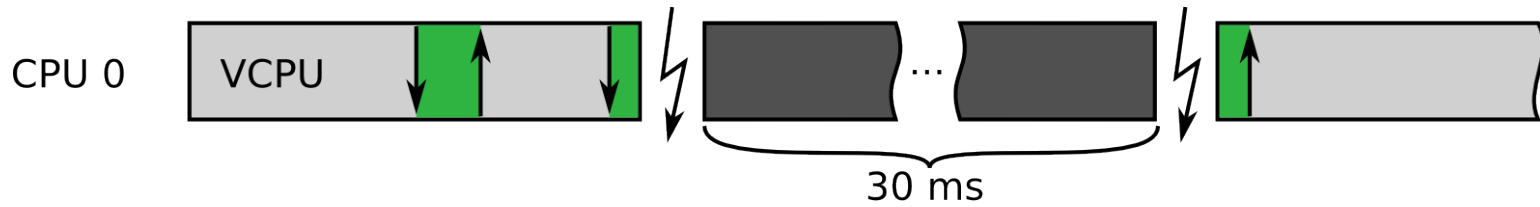
Spinlocks and Virtualization



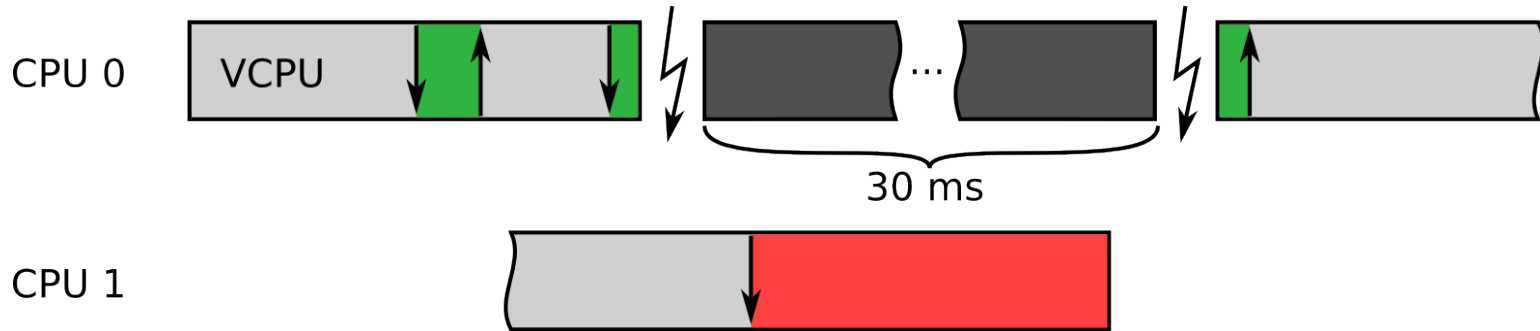
Spinlocks and Virtualization



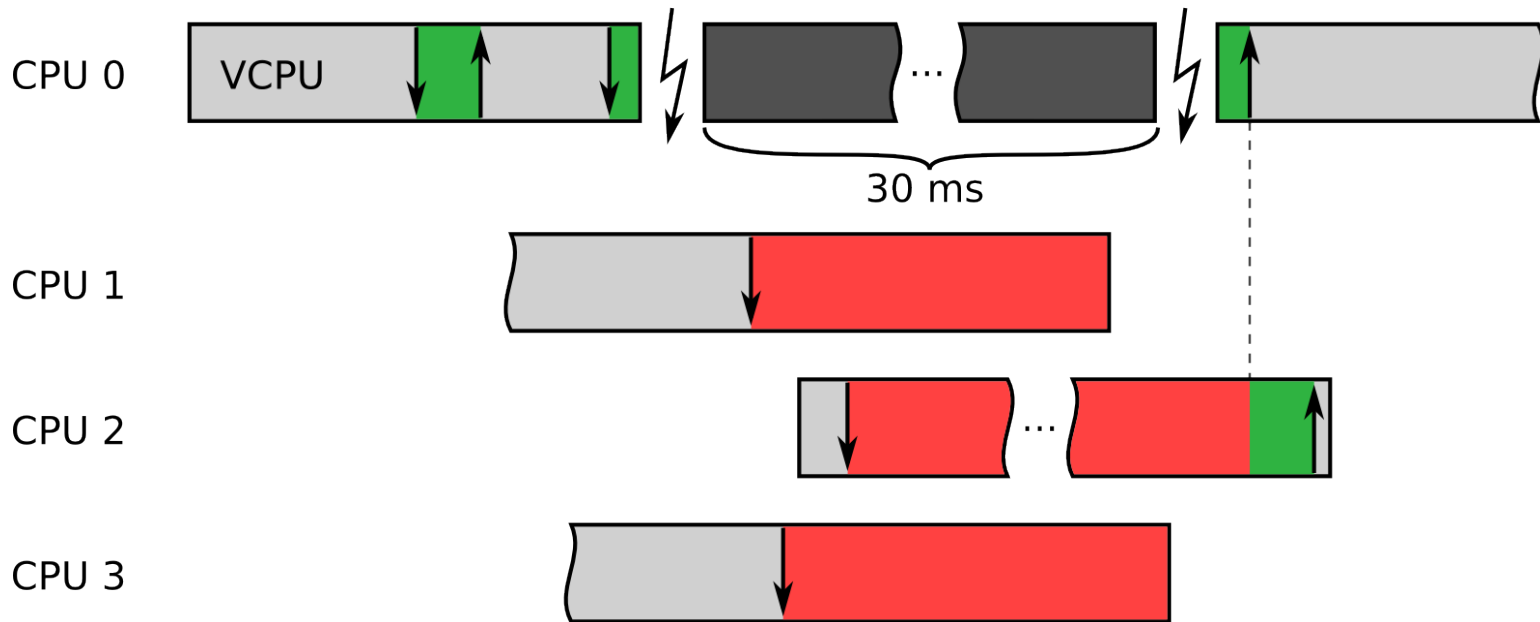
Spinlocks and Virtualization



Spinlocks and Virtualization

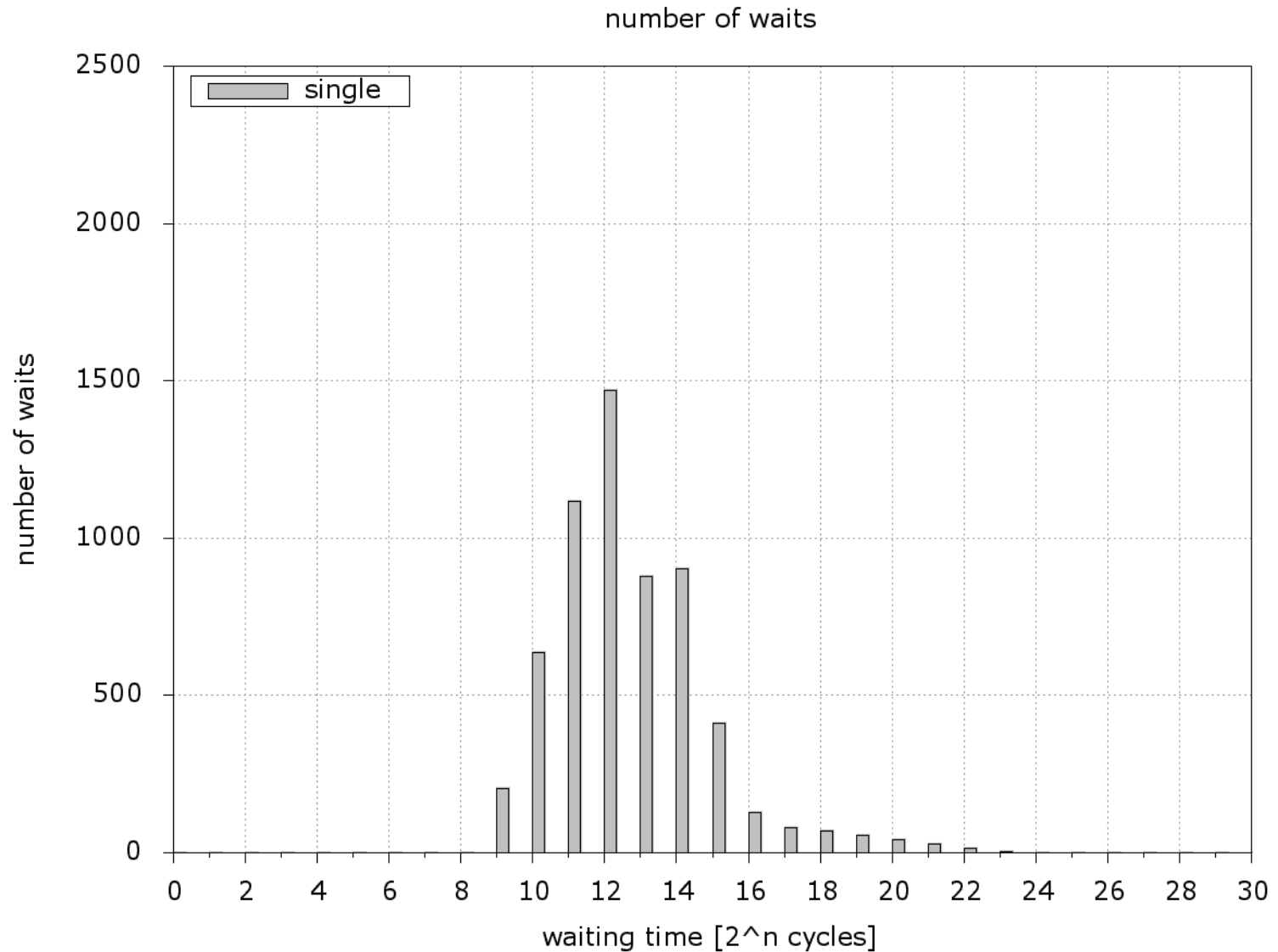


Spinlocks and Virtualization

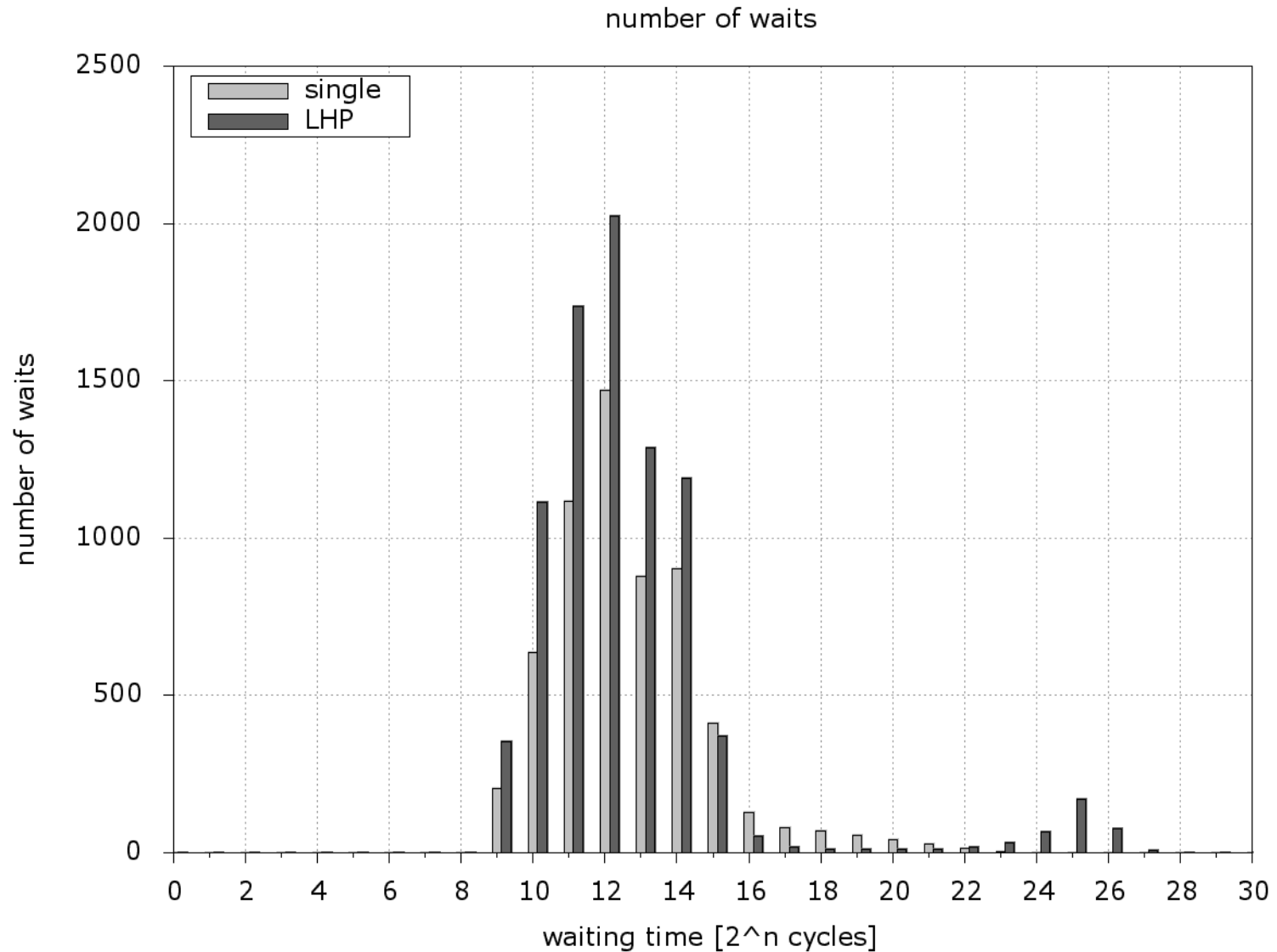


Is lock-holder preemption problematic?

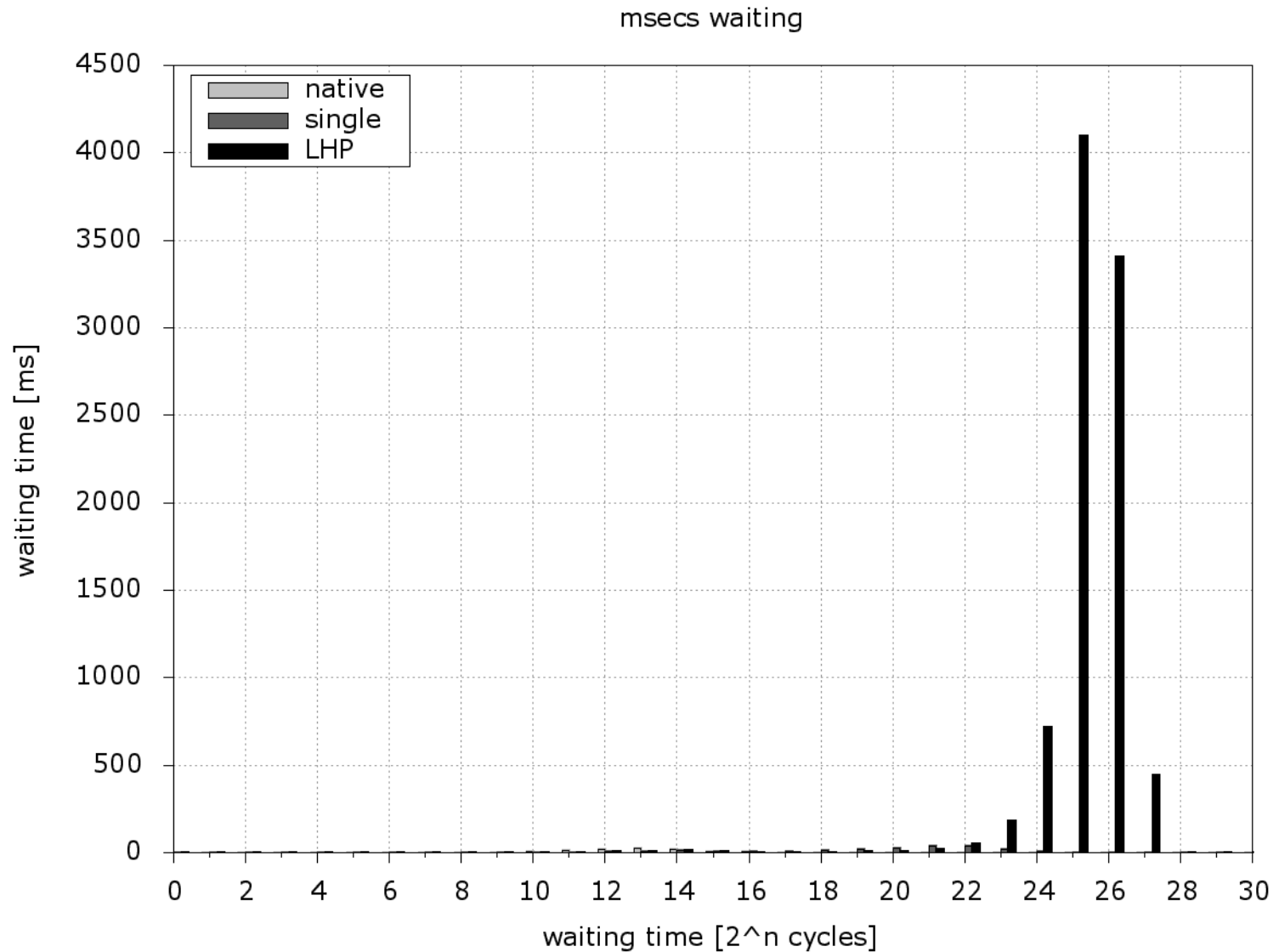
Kernbench in a Guest



Kernbench vs. 'while(true)'



Time, not Times



And in Numbers?

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

What can we do about it?

Dealing with lock-holder preemption

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

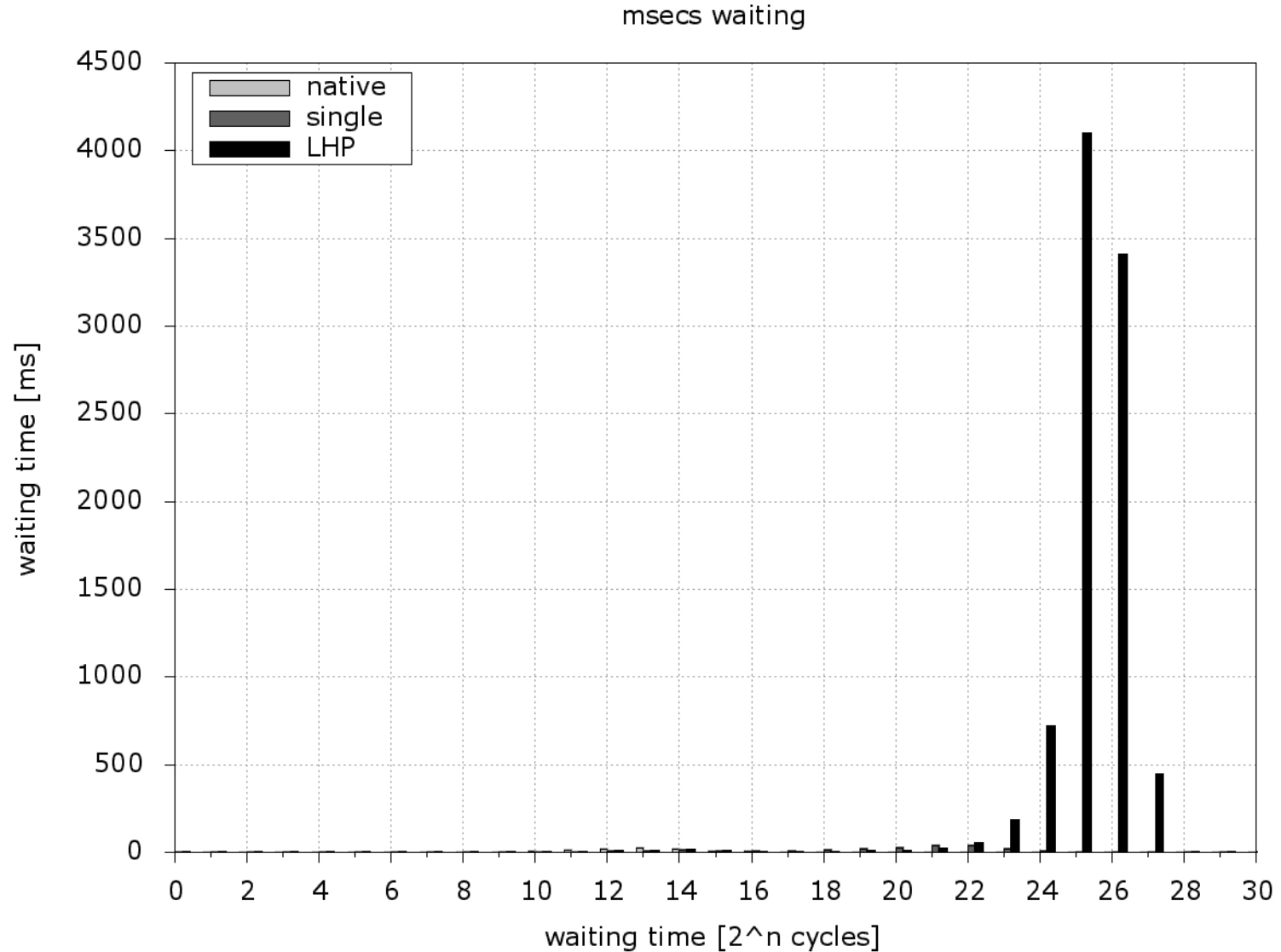
Helping locks: Ingredients

- 1) Guest kernel: new 'yield' hypercall when waiting unusually long
 - Modify spinlock loop

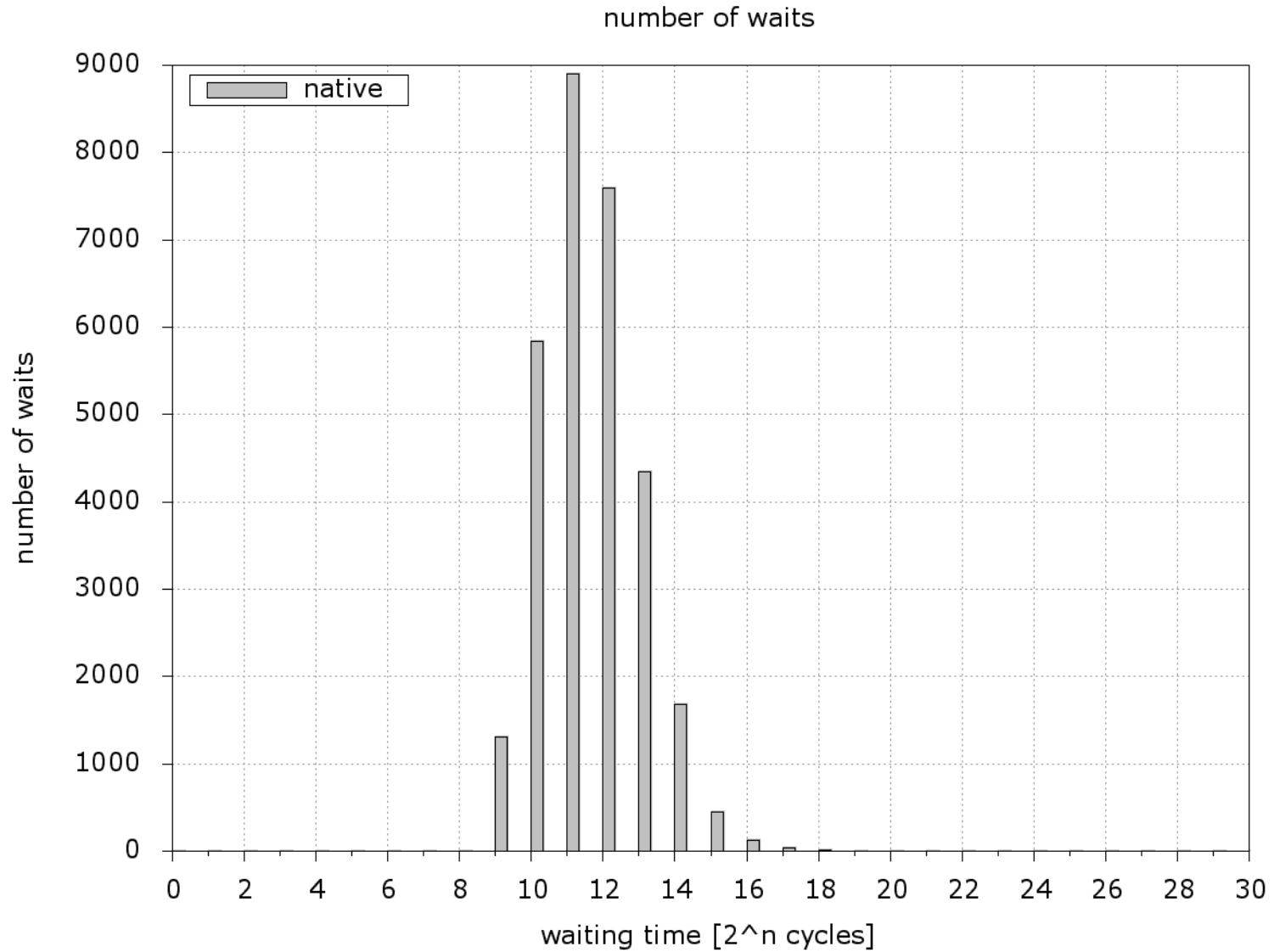
- 2) Reasonable threshold for 'unusually long'
 - Histograms help

- 3) Selecting which VCPU to switch to

Threshold: Upper boundary



Threshold: Lower boundary



Scheduling Strategy

Good choices:

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

Neither/nor:

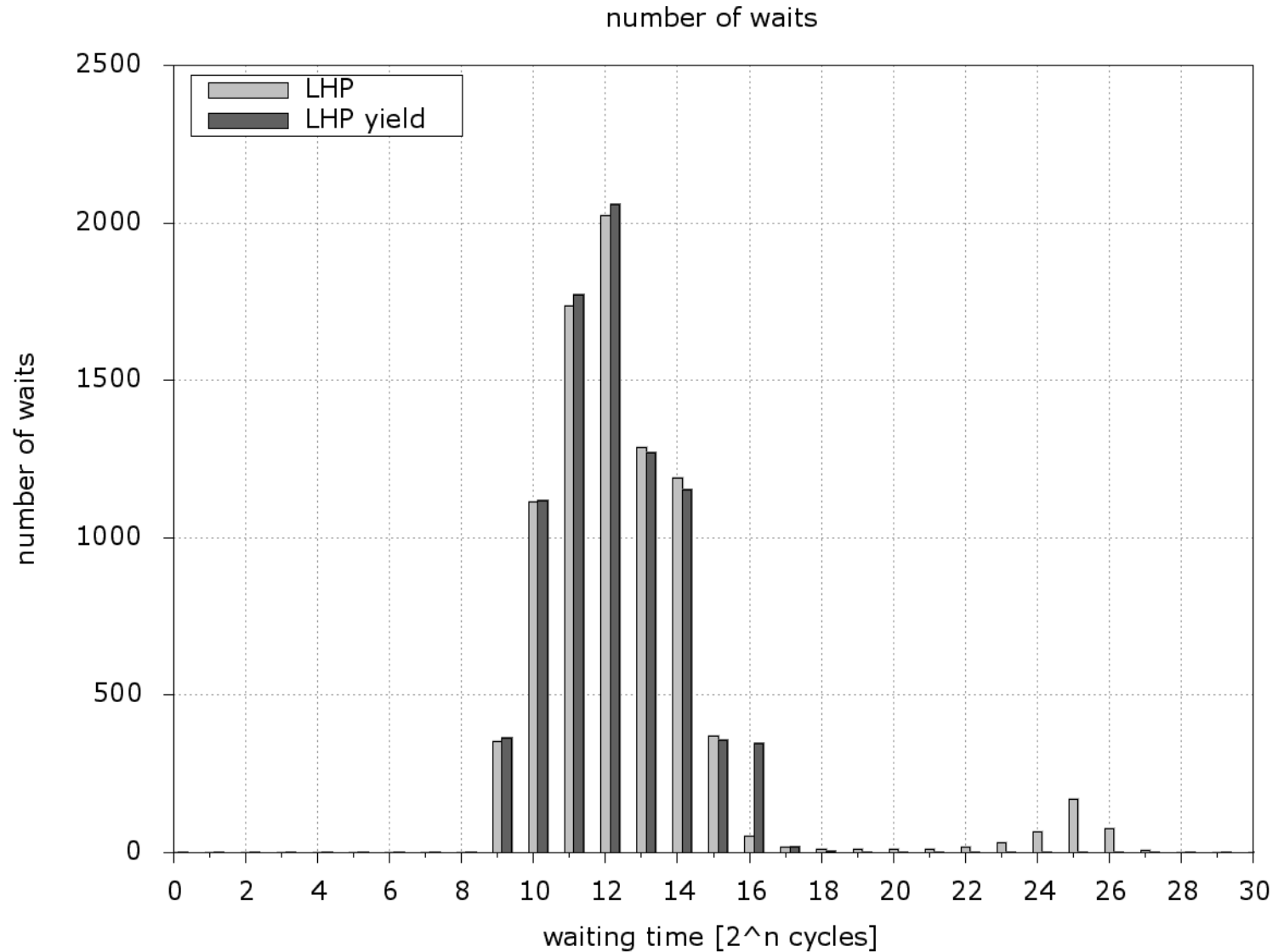
- VCPUs in user space

Bad choices:

- VCPUs which yielded recently

What about performance?

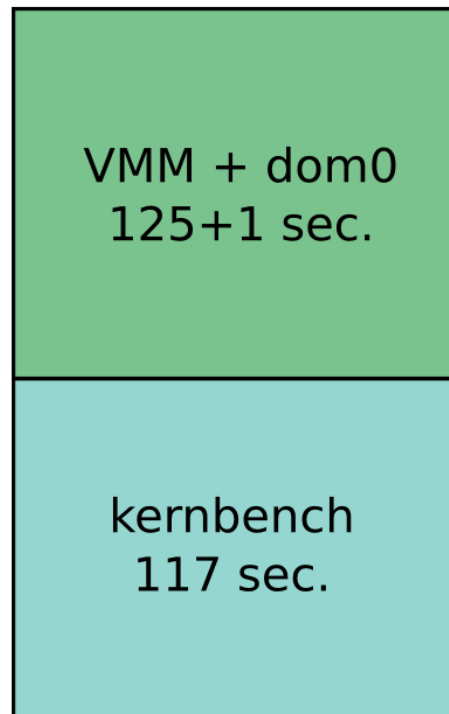
Histogram with 'yield' hypercall

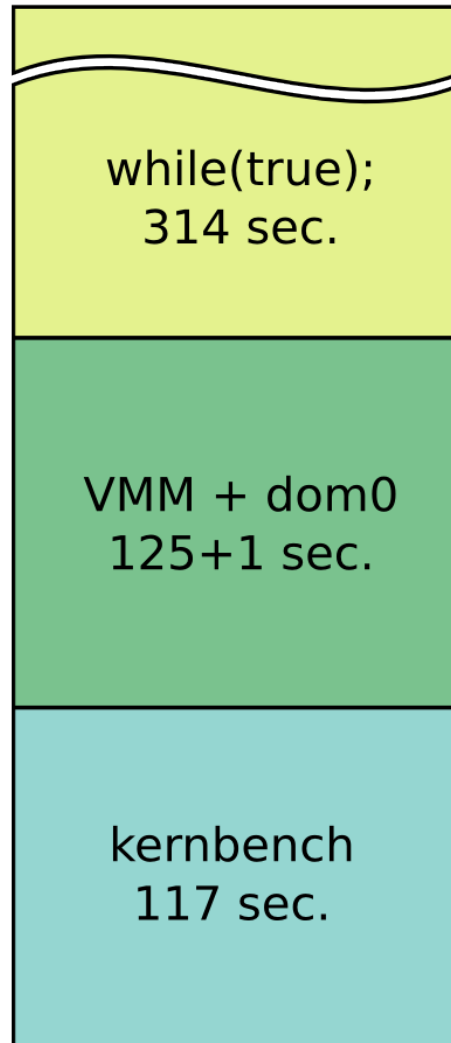


Performance

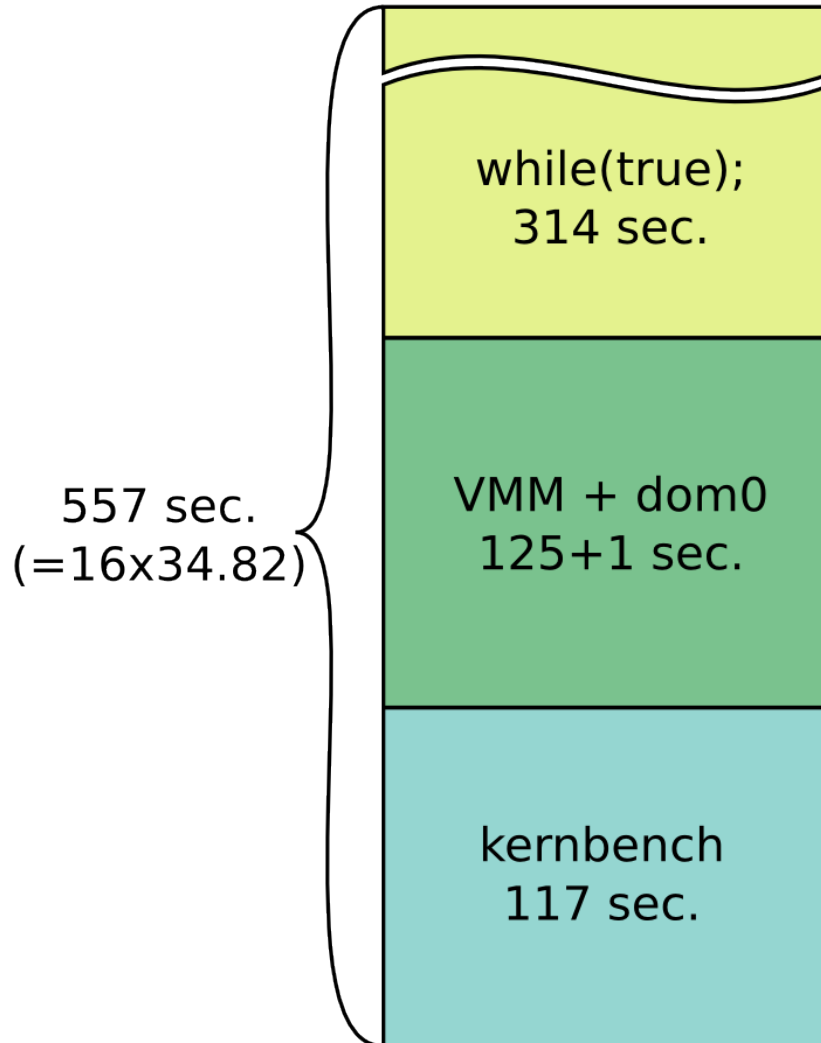
	wall clock [s]	guest time [s]	time spent spinning [s]	spinning [%]
LHP	34.8	117.3	9.0	7.6%
yield	33.5	108.4	0.0	0.0%
difference	-3.9%	-7.6%		-7.6%

kernbench
117 sec.

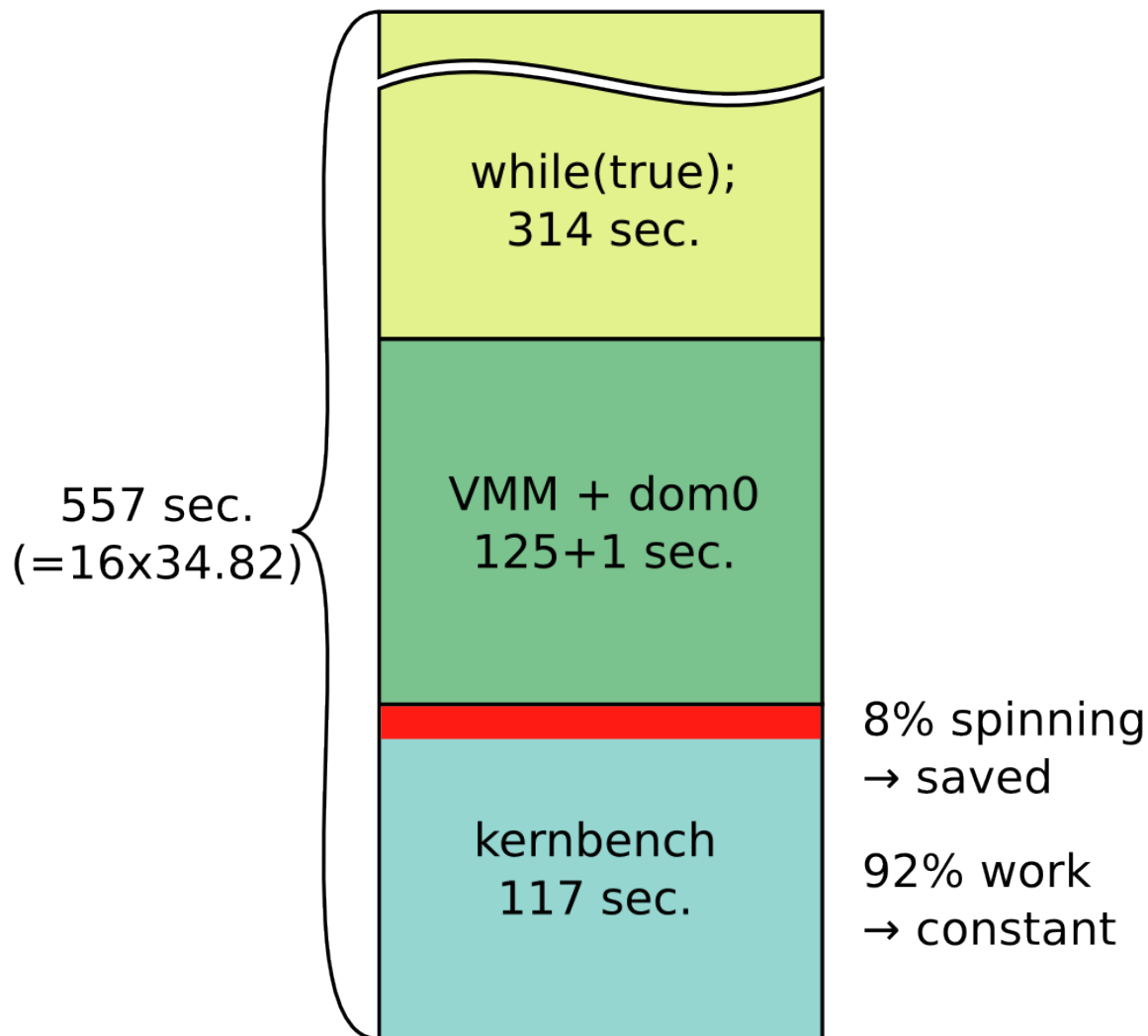




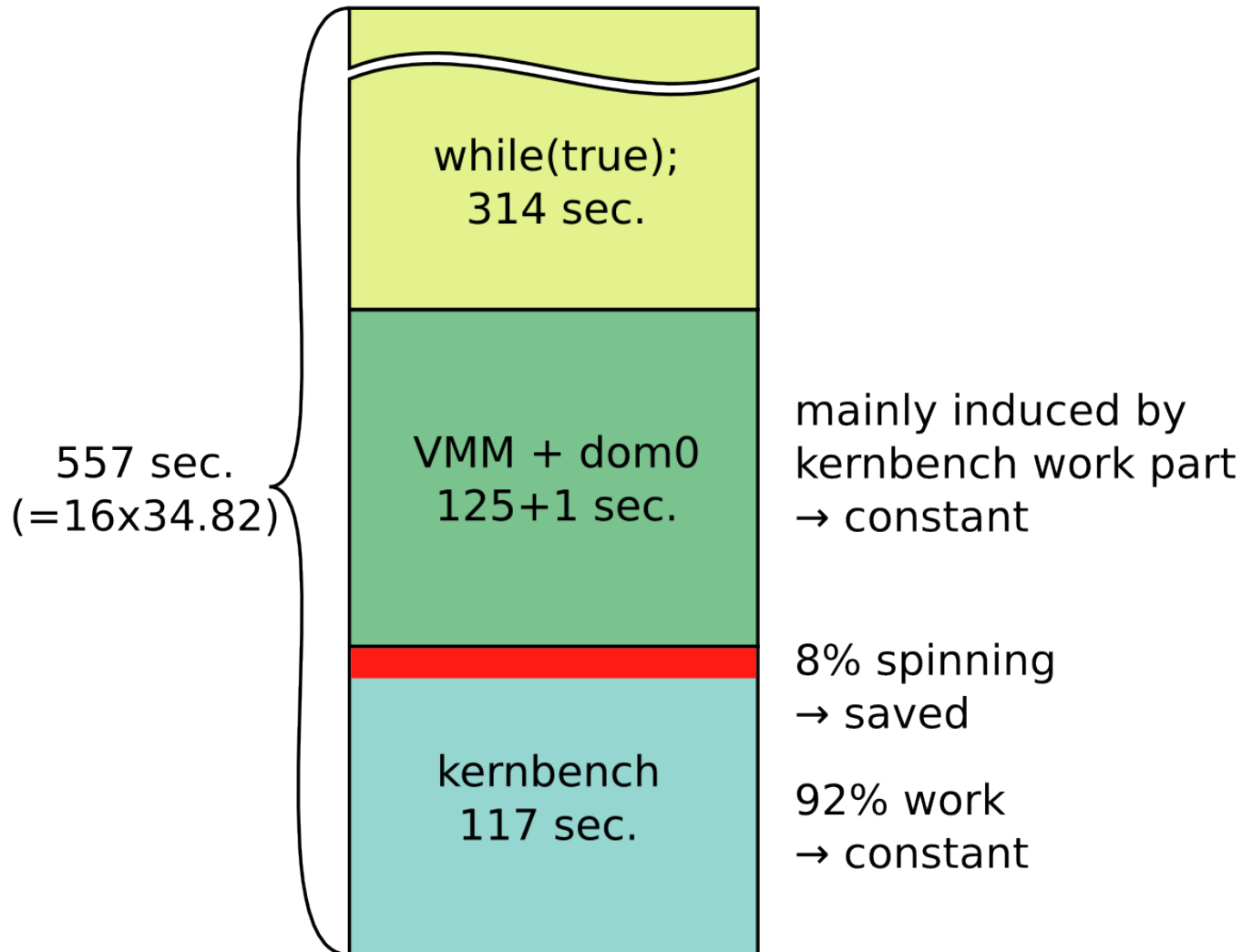
Efficiency



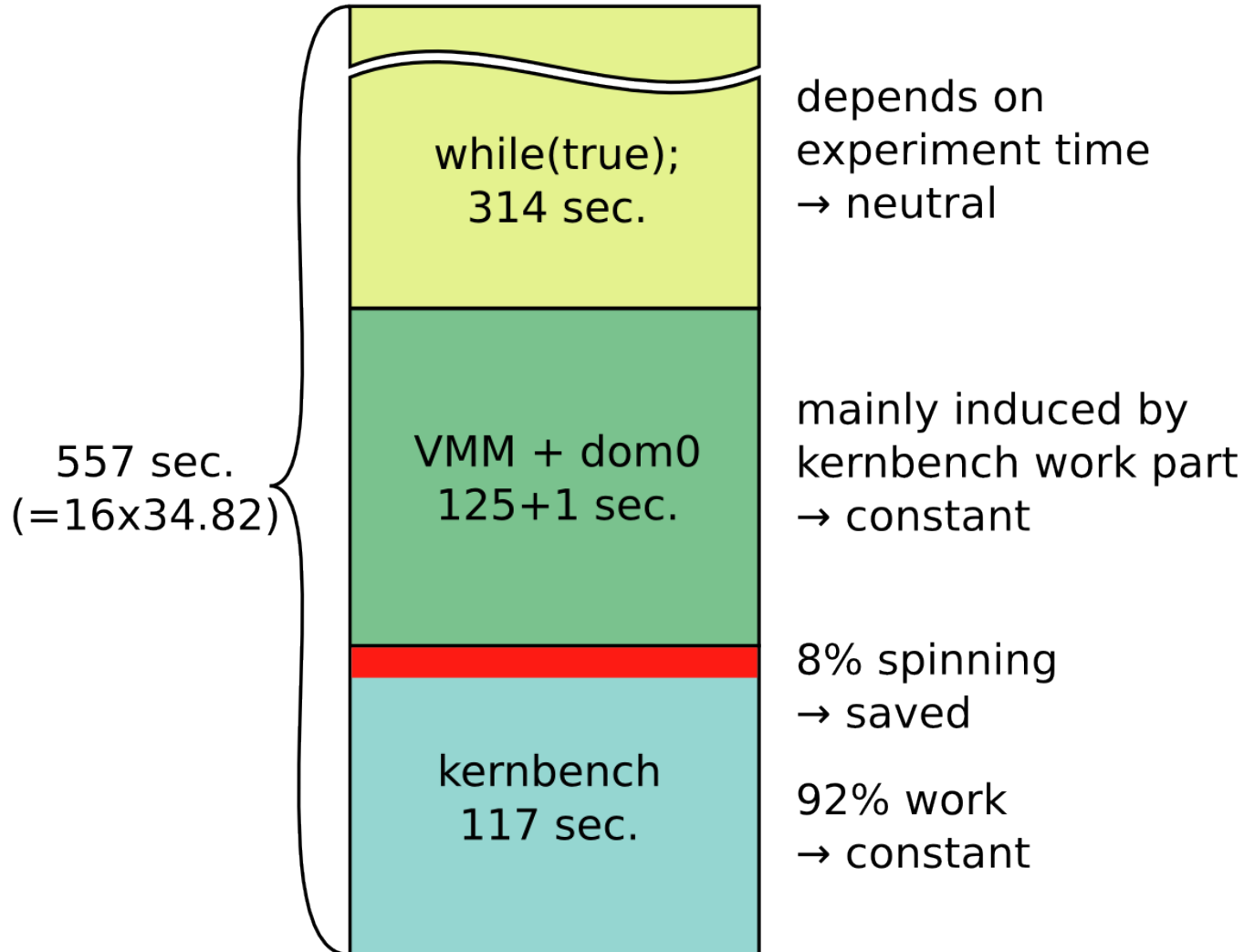
Efficiency



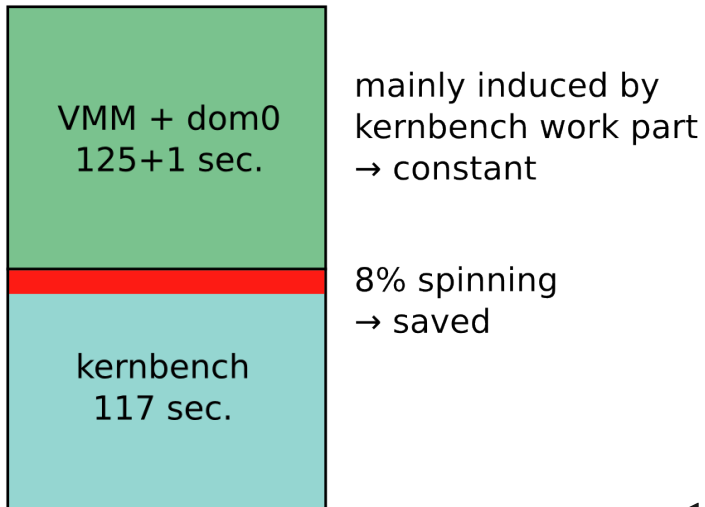
Efficiency



Efficiency



Efficiency



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

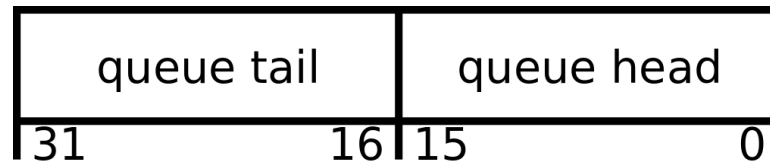
- Real result of 3.9% is reasonable
- Highly efficient

FIFO ticket spinlocks

FIFO ticket spinlocks

Next ticket in dispenser: queue tail

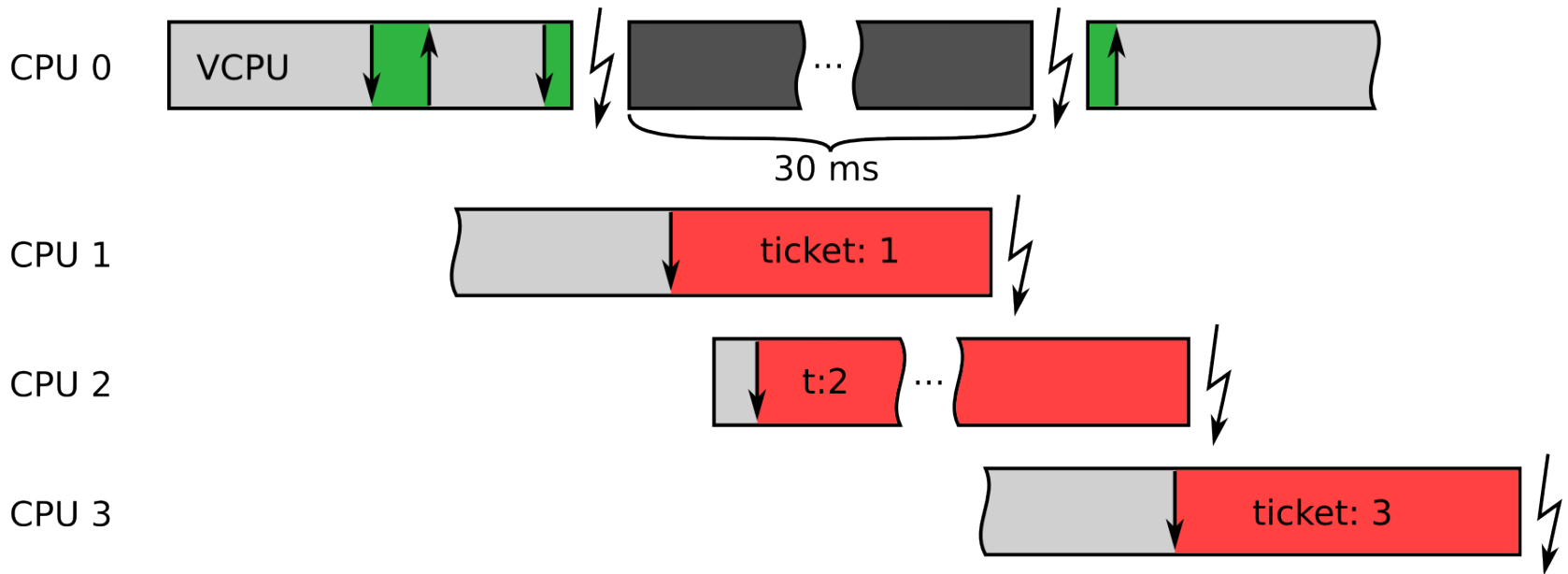
„Now serving“ display at counter: queue head



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]	spinning [%]
LHP	2825.1	22434.2	22270.4	99.3%

Ticket locks and virtualization

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

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

Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

©2008 Advanced Micro Devices, Inc. All rights reserved.