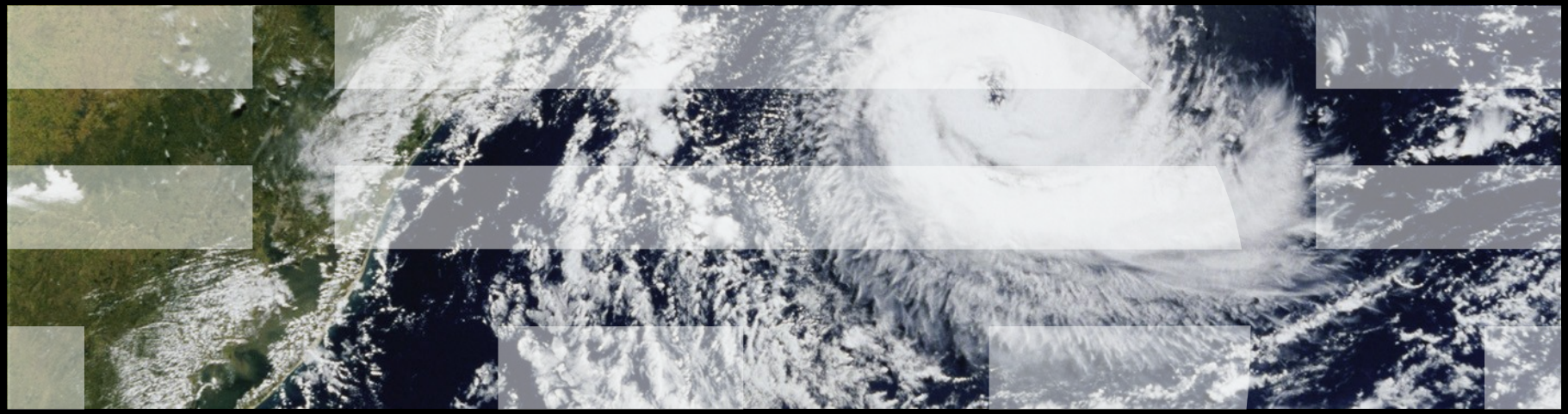
Paul E. McKenney, IBM Distinguished Engineer, Linux Technology Center
Member, IBM Academy of Technology
Linux Plumbers Conference, New Orleans, LA, USA September 19, 2013

Improving Energy Efficiency On Asymmetric Multiprocessing Systems

Work done while assigned to Linaro,
Joint work with Dietmar Eggeman and Robin Randhawa (ARM Ltd.)



© 2013 IBM Corporation

Overview

- ARM big.LITTLE Architecture
- ARM big.LITTLE Energy-Efficiency Strategy
- Review: Morten Rasmussen Approach
- RCU and big.LITTLE Energy Efficiency

© 2013 IBM Corporation

ARM big.LITTLE Architecture

© 2013 IBM Corporation

What is big.LITTLE?

- "Continuation of voltage/frequency scaling by other means"
  - Low voltage and frequency: Per-transistor leakage dominates power
  - Therefore need to reduce the number of transistors
  - But need the full complement of transistors for good performance
- Solution: Have two sets of CPUs
  - Cortex A15: "big" CPUs with deep pipelines and many functional units
    - Optimized for performance
  - Cortex A7: "LITTLE" CPUs with short pipelines and few functional units
    - Optimized for energy efficiency
  - Get high performance *and* good energy efficiency
- Two basic configurations:
  - big.LITTLE switcher
  - big.LITTLE MP (the focus of this talk)

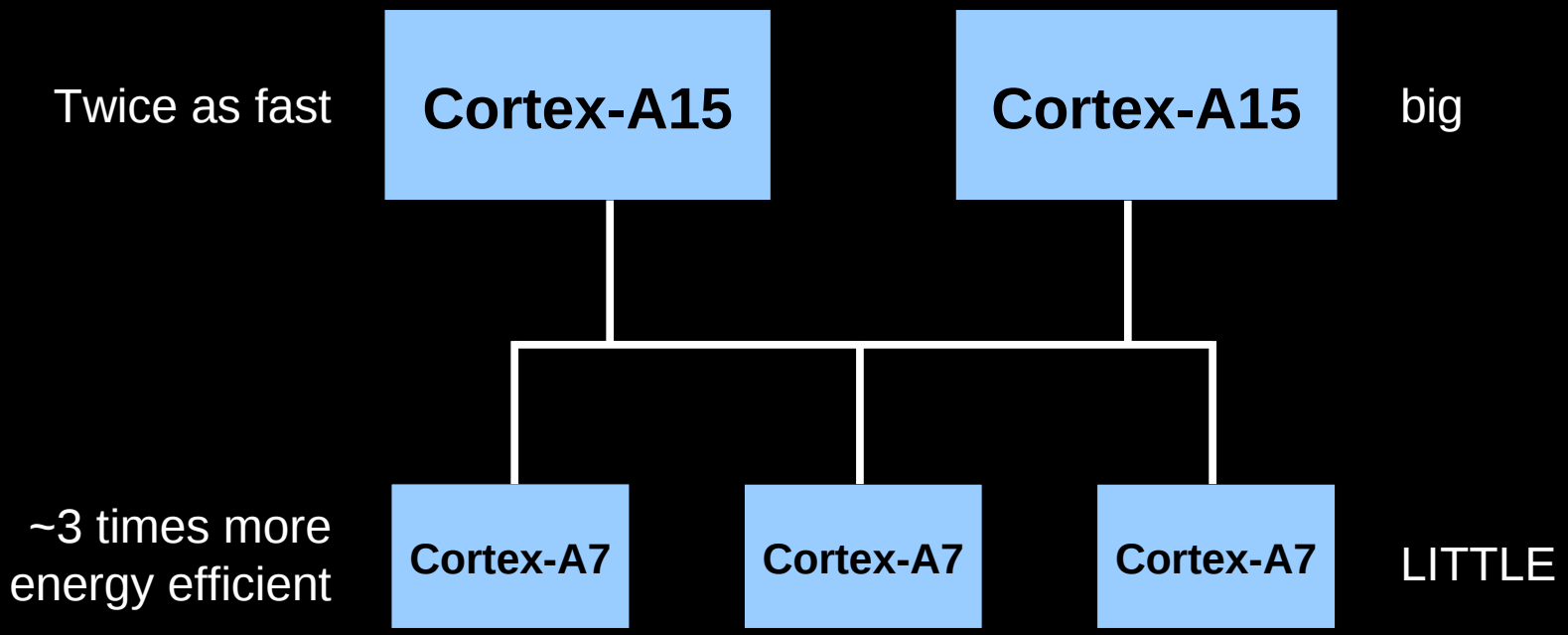
© 2013 IBM Corporation

What is big.LITTLE MP?

- Each physical CPU, whether big or LITTLE, is separately visible to kernel and to applications
- Allows any combination of big and LITTLE CPUs to be used concurrently, but requires more awareness of big.LITTLE
  - ~~Emulation: Need software before hardware is available~~
  - Applications: Need application awareness until kernel support
  - Test workloads: Need to enable tests without big.LITTLE hardware
  - CPU hotplug: There will be corner cases...
  - Scheduler: Automation of CPU choice is the ideal

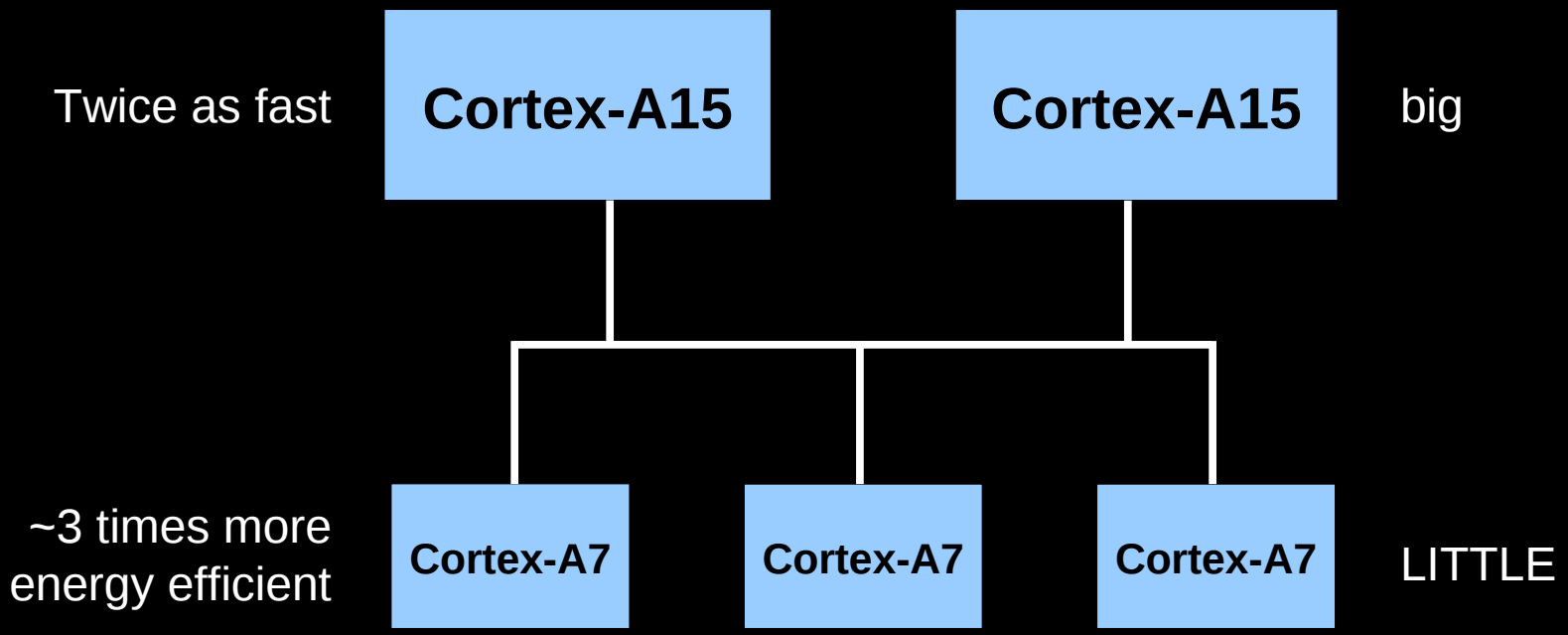
© 2013 IBM Corporation

ARM big.LITTLE Architecture



© 2013 IBM Corporation

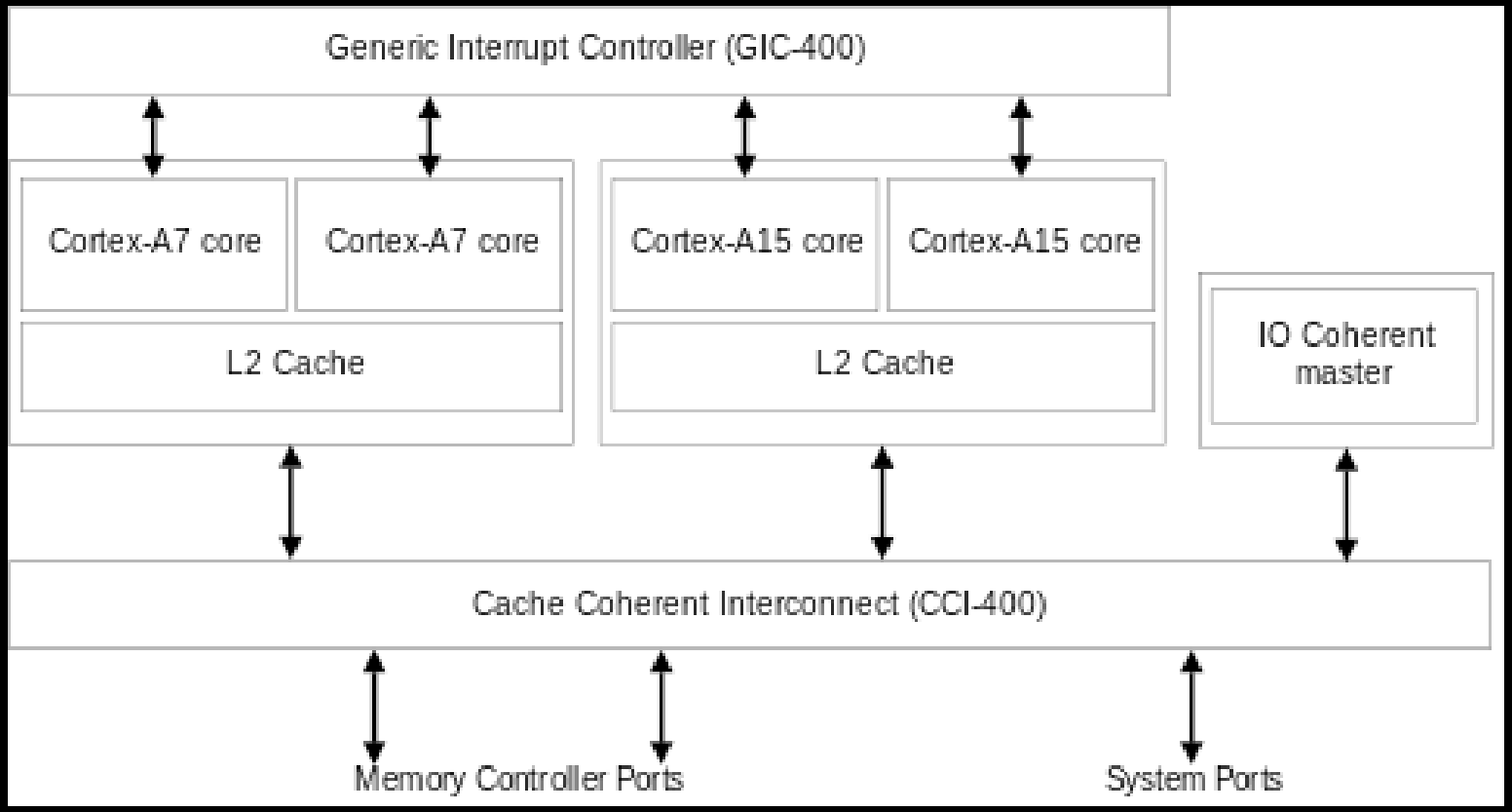
ARM big.LITTLE Architecture



Unfortunately, the Linux kernel assumes all CPUs are similar...

© 2013 IBM Corporation

ARM big.LITTLE Schematic



© 2013 IBM Corporation

ARM big.LITTLE Energy-Efficiency Strategy

© 2013 IBM Corporation

ARM big.LITTLE Energy-Efficiency Strategy

- Run on the LITTLE by default
- Run on big if heavy processing power is required
  - Power down big CPUs when not needed
- In other words, if feasible, run on LITTLE for efficiency, but run on big if necessary to preserve user experience
  - Use big CPUs for media processing, rendering, etc.
  - This suggests that RCU callbacks should run on LITTLE CPUs, possibly also for timers and other low-priority asynchronous events
  - Key point: Goal of big.LITTLE scheduling is to distribute tasks unevenly to handle different energy-efficiency and performance goals
  - Unlike traditional SMP, it now matters where a task is scheduled

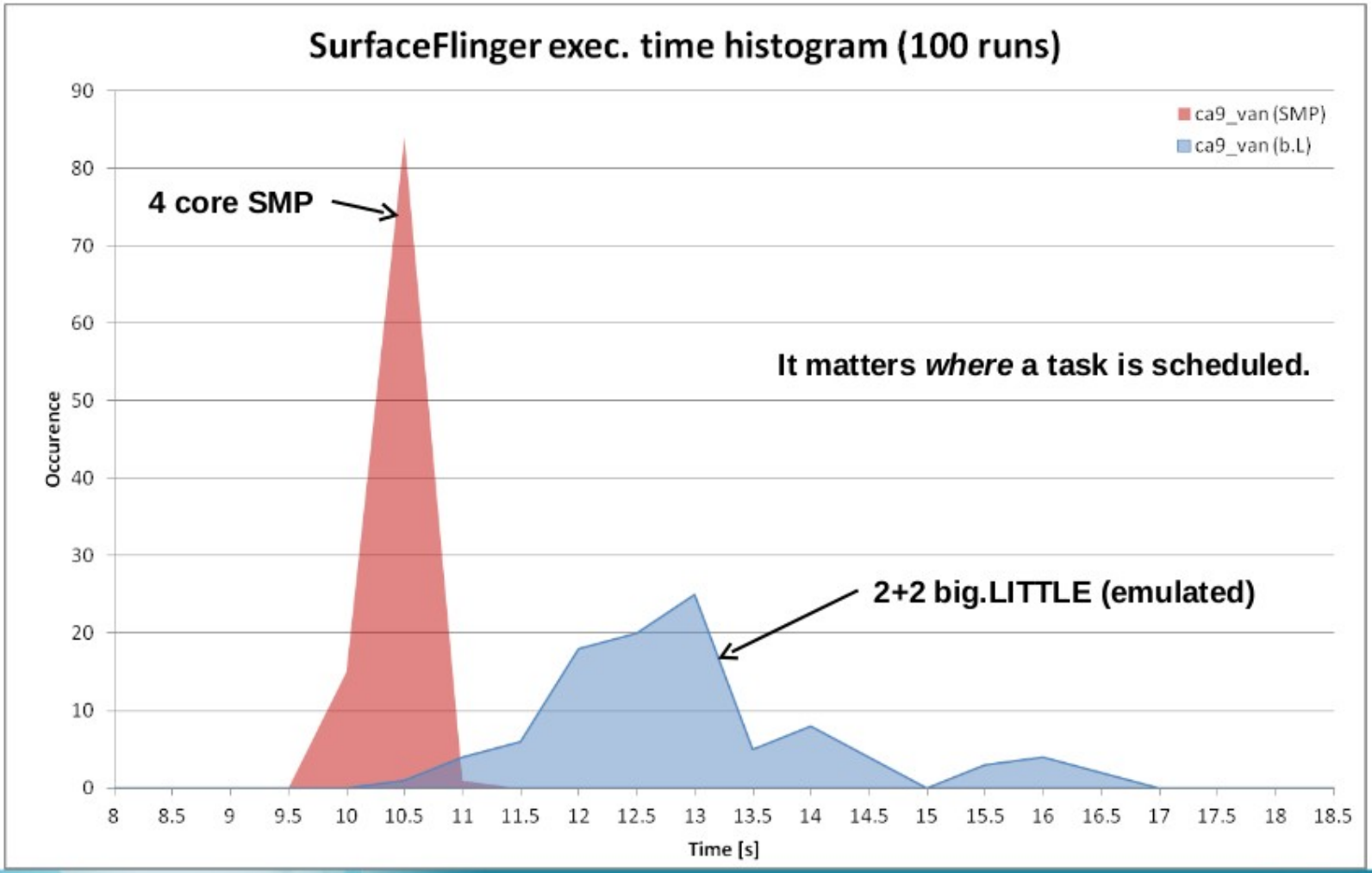
© 2013 IBM Corporation

Review: Morten Rasmussen Approach

© 2013 IBM Corporation

Without Morten Rasmussen Approach: Bad!!!

- Example: Android UI render thread execution time.



The Architecture for the Digital World®
ARM®

© 2013 IBM Corporation

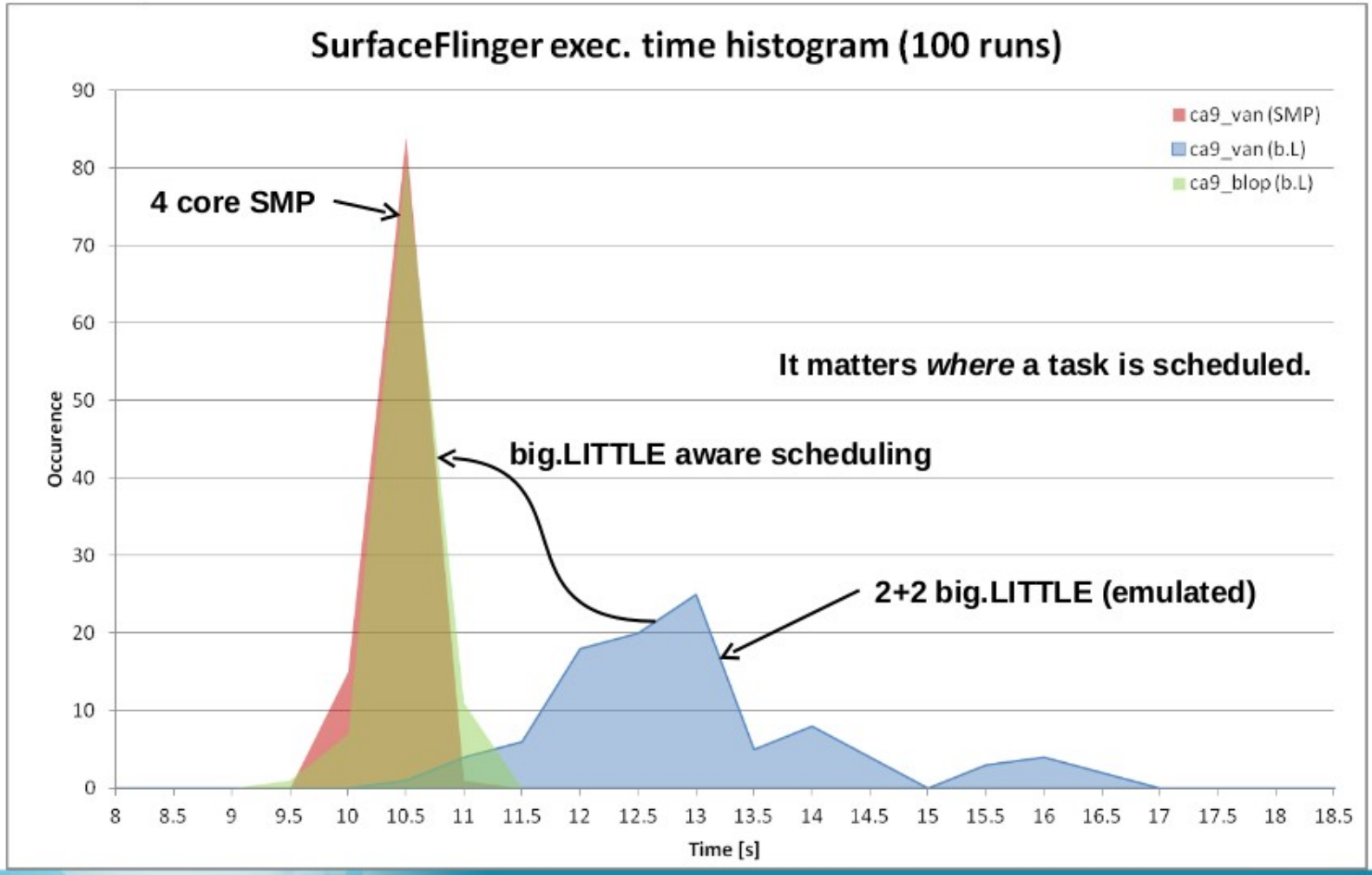
Review: Morten Rasmussen Approach

- Based on Paul Turner's entity load-tracking patches
- Strategy: Run all tasks on LITTLE cores unless:
  - The task load is above a fixed threshold, and
  - The task priority is default or higher
- Implementation:
  - Set up big and LITTLE sched domains without load balancing
  - When long-running high-priority task awakens, run it on a big core.
  - Periodically check for high-priority tasks transitioning into long-running mode, and migrate them to big CPUs
- Performance results rival those on a system with all big cores
  - (See next slide)

© 2013 IBM Corporation

With Morten Rasmussen Approach: Pretty Good!

- Example: Android UI render thread execution time.



The Architecture for the Digital World®

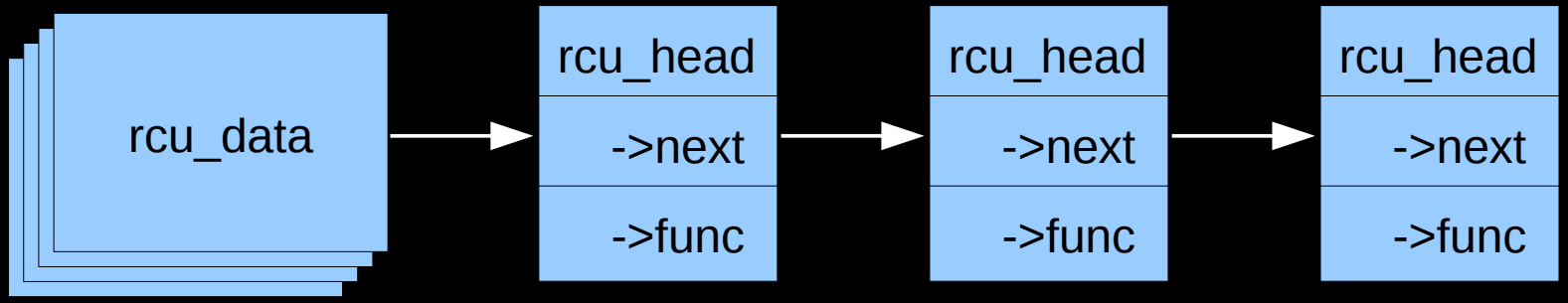
© 2013 IBM Corporation

RCU and big.LITTLE Energy Efficiency

© 2013 IBM Corporation

What is RCU? (AKA Read-Copy Update)

- For an overview, see http://lwn.net/Articles/262464/
- For the purposes of this presentation, think of RCU as something that defers work, with one work item per callback
  - Each callback has a function pointer and an argument
  - Callbacks are queued on per-CPU lists, invoked after “grace period”
  - Deferring the work a bit longer than needed is OK, deferring too long is bad (splat after 20 seconds) – but failing to defer long enough is fatal
  - RCU allows extremely fast & scalable read-side access to shared data



© 2013 IBM Corporation

RCU:
Tapping The Awesome Power of Procrastination For Two Decades!!!

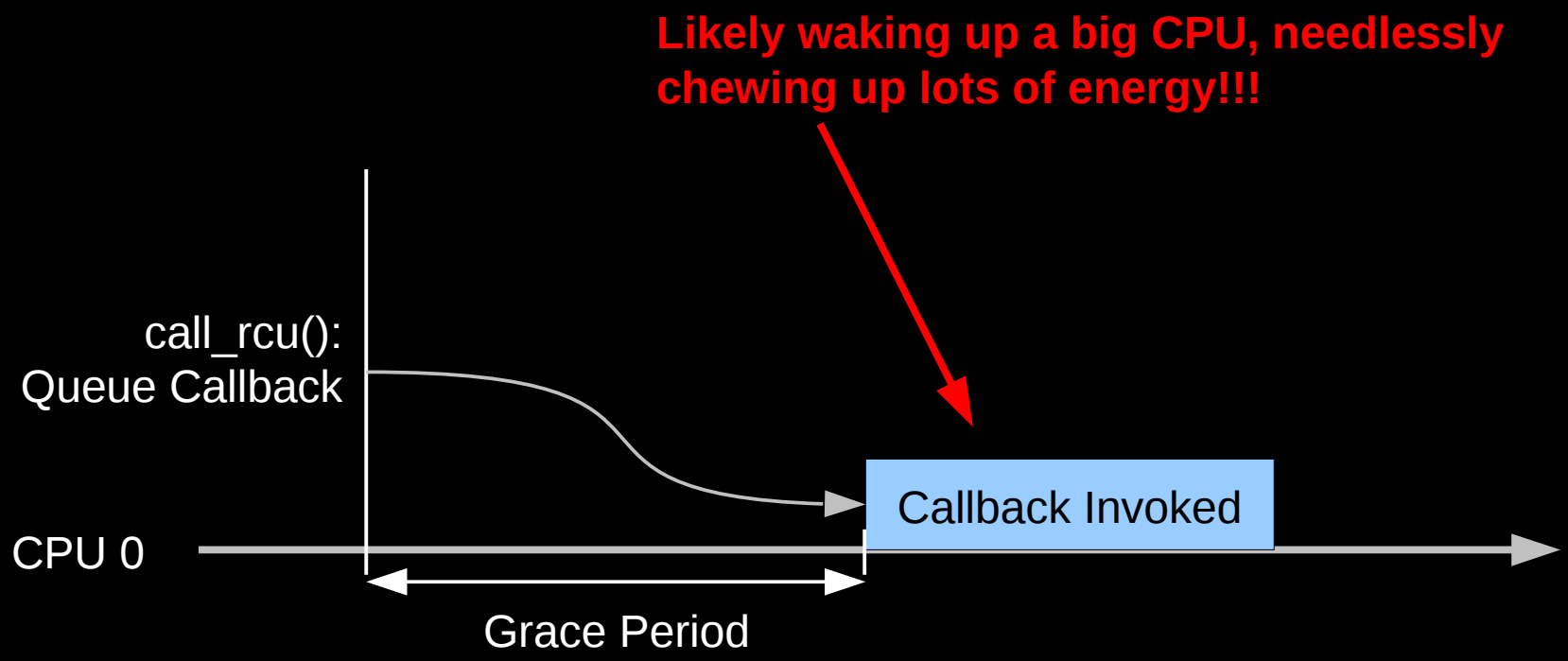
© 2013 IBM Corporation

RCU:
Tapping The Awesome Power of Procrastination For Two Decades!!!

But Procrastination has a Dark Side...

© 2013 IBM Corporation

Procrastination's Dark Side: Eventually Must Do Work



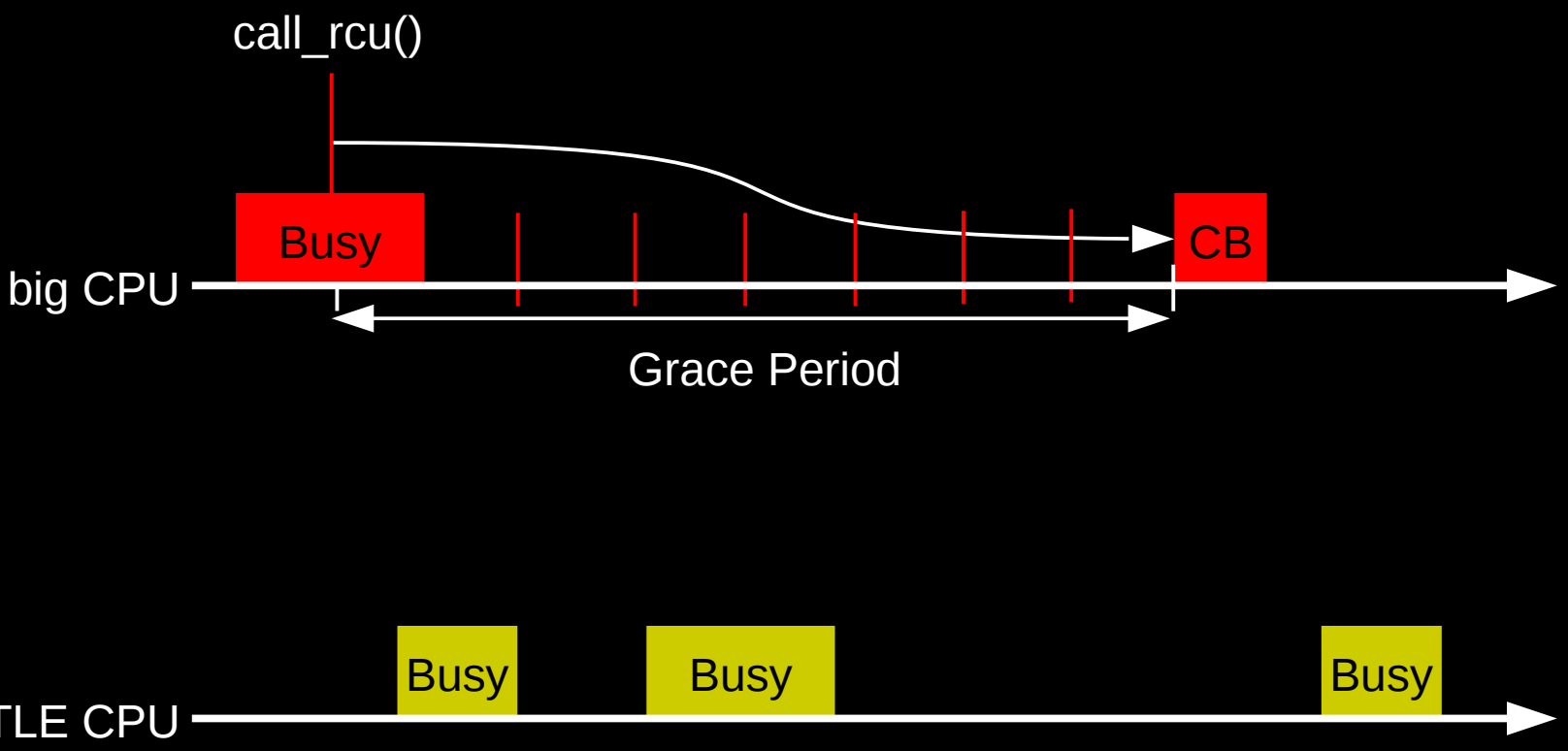
© 2013 IBM Corporation

Two Ways Of Conserving Energy

- Offload RCU callbacks to LITTLE CPUs
- Use RCU_FAST_NO_HZ to reduce wakeups

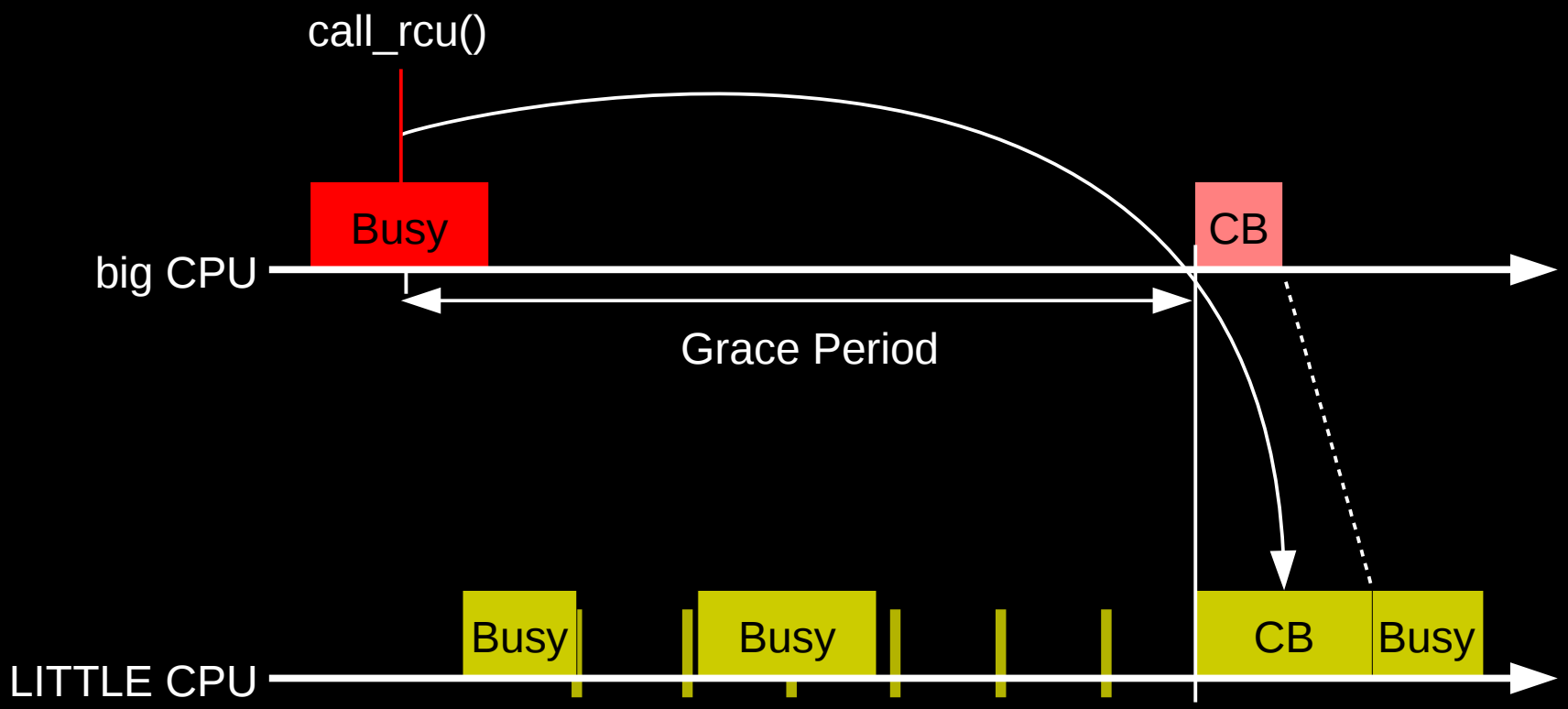
© 2013 IBM Corporation

Base Case



© 2013 IBM Corporation

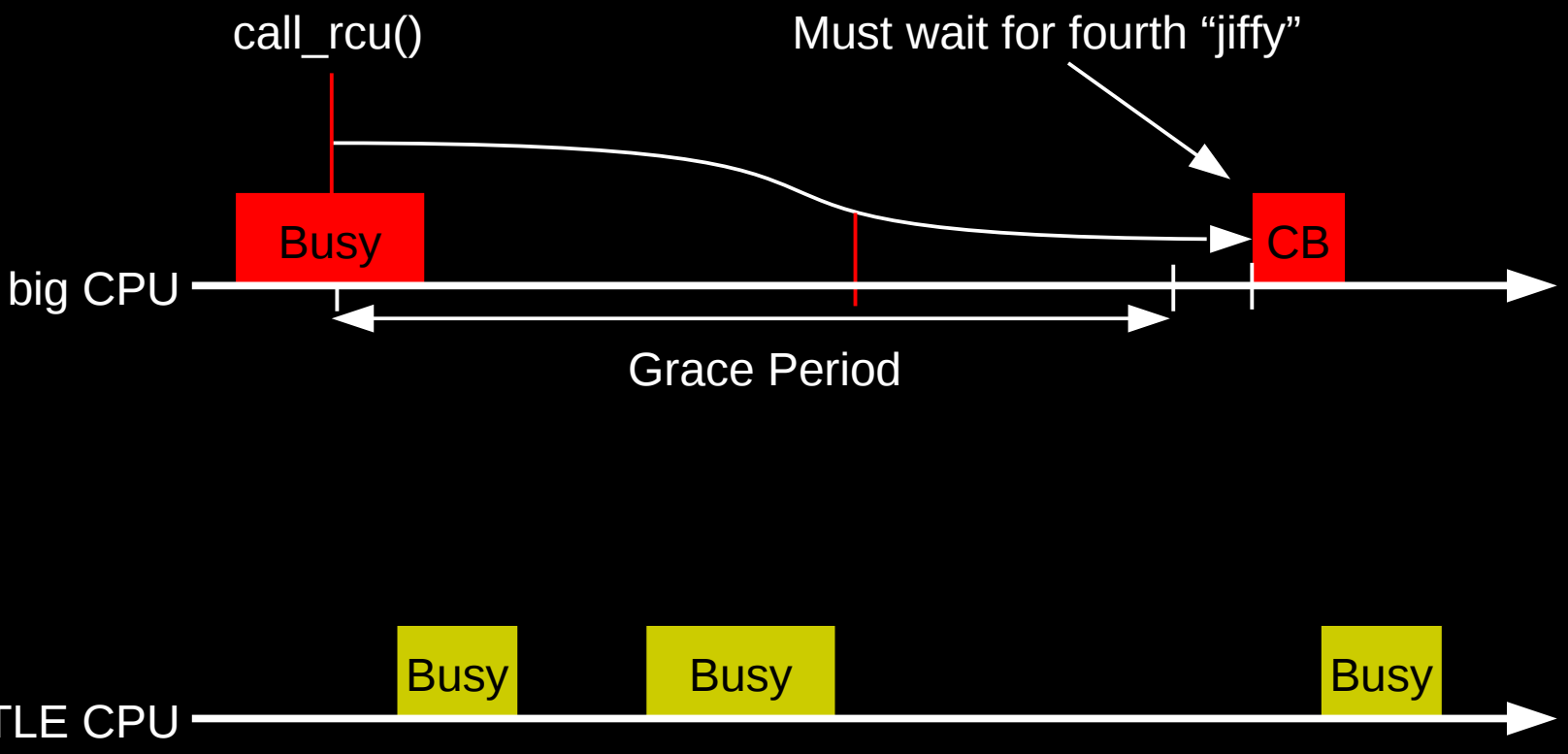
1: ARM big.LITTLE With RCU Callback Offloading



Slower...
But 3x better energy efficiency

© 2013 IBM Corporation

1: ARM big.LITTLE With Reduced Wakeups



© 2013 IBM Corporation

Results Summary

	Reference		RCU Processing Offload				Enforced Idle			
Benchmark	E (J)	T (s)	Energy	Benefit	Time	Benefit	Energy	Benefit	Time	Benefit
cyclictest	1.75	3.13	1.47	15.98%	3.23	**-3.38%**	1.47	**16.13%**	3.13	**-0.07%**
sysbench	32.68	9.14	31.61	3.29%	8.99	**1.68%**	31.12	4.77%	8.99	**1.70%**
andebench2	59.51	20.54	57.91	2.70%	20.37	**0.83%**	57.99	**2.57%**	21.59	**-5.09%**
andebench8	174.04	45.87	170.37	**2.11%**	46.19	**-0.70%**	166.91	4.09%	46.60	**-1.59%**
audio	7.87	30.01	7.39	**6.05%**	30.03	**-0.04%**	6.28	20.16%	30.02	**-0.04%**
audio+bbench	99.05	156.29	93.02	6.09%	155.58	**0.45%**	86.95	12.21%	160.75	**-2.85%**

Actual energy measurements taken on real hardware.

© 2013 IBM Corporation

Summary: Which is Better?

- Both produce real benefits
  - Offloading gives slightly better wall-clock time
  - Enforced idle gives slightly better energy efficiency
  - Combining them does not help
- Both are needed
  - Offloading for real time and reduced OS jitter
  - Enforced idle for SMP energy efficiency
- Offloading other deferred operations may be helpful
  - Timers, workqueues, ...

© 2013 IBM Corporation

Legal Statement

- This work represents the view of the author and does not necessarily represent the view of IBM.
- IBM and IBM (logo) are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.

© 2013 IBM Corporation

Questions?

© 2013 IBM Corporation