HITACHI
Inspire the Next

Evaluation of Real-time Property in Embedded Linux

LinuxCon Japan 2014

Hiraku Toyooka, Hitachi
<hiraku.toyooka.gu@hitachi.com>

Linux Technology Center
Yokohama Research Lab.

whoami

HITACHI
Inspire the Next

- Hiraku Toyooka
  - Software engineer at Hitachi

- Working on operating systems
  - Linux (mainly) for industrial control systems
    - Server, Embedded
    - Tracing
    - Real-time virtualization (partitioning)

Linux Technology Center
Yokohama Research Lab.

Agenda

HITACHI
Inspire the Next

- What is a real-time system?
- Real-time features in Linux
- Tuning points for evaluation
- Performance measurement

Linux Technology Center
Yokohama Research Lab.

HITACHI
Inspire the Next

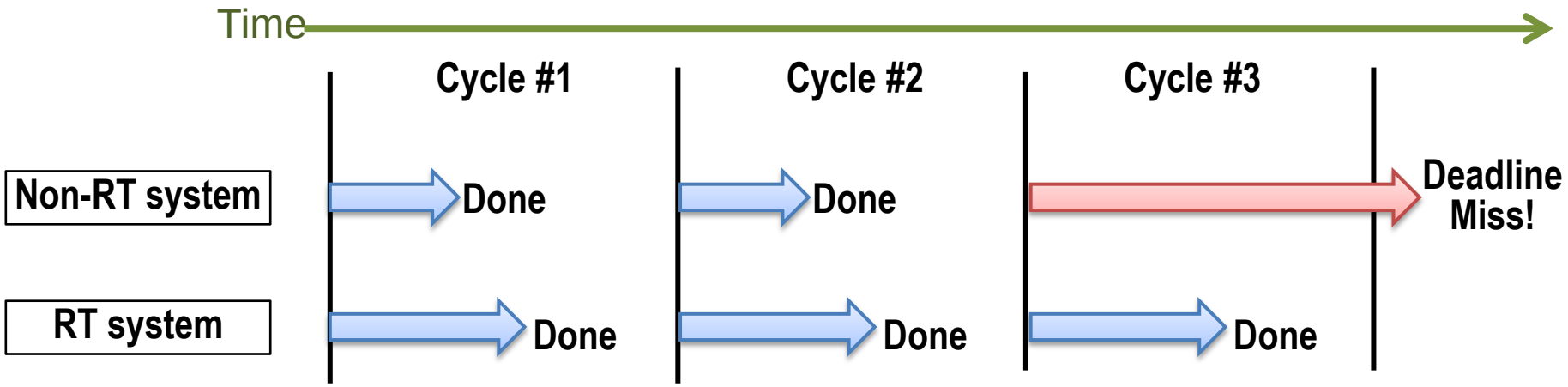
What is a real-time system?

Linux Technology Center
Yokohama Research Lab.

What is a real-time system?

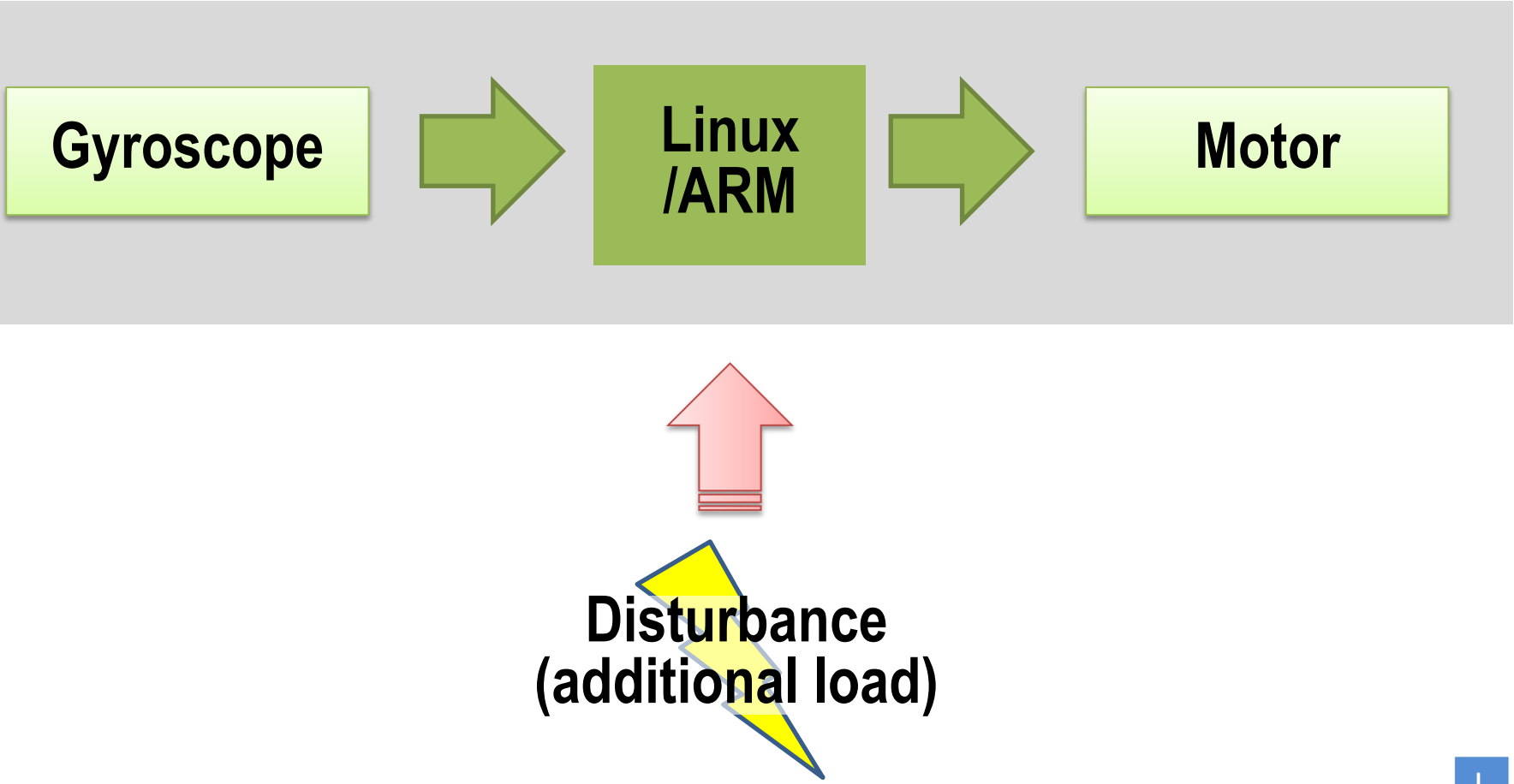
HITACHI
Inspire the Next

- Real-time system must always handle events within deadlines
  - Real-time doesn't mean fast
  - Worst case performance is important
  - Deterministic behavior is required



HITACHI
Inspire the Next

- Robot control (inverted pendulum)



HITACHI
Inspire the Next

Real-time features in Linux

Linux Technology Center
Yokohama Research Lab.

Linux and real-time

HITACHI
Inspire the Next

- Linux is originally designed to be a general purpose OS (GPOS)
  - High functionality
  - Average or peak performance is usually important for GPOS
  - Latency varies widely

Linux Technology Center
Yokohama Research Lab.

Linux and real-time

HITACHI
Inspire the Next

- Current mainline Linux has a variety of real-time features
  - Certain level of deterministic behavior by appropriate configurations and tuning

- PREEMPT_RT patchset reduce maximum latency much more
  - Many real-time features in mainline are derived from PREEMPT_RT

(Some) Real-time features

HITACHI
Inspire the Next

- Mainline
  - Fixed-priority scheduling
  - Kernel preemption
  - High resolution timer (hrtimer)
  - IRQ thread
  - Others
- Out of tree
  - PREEMPT_RT patchset

Fixed-priority scheduling

HITACHI
Inspire the Next

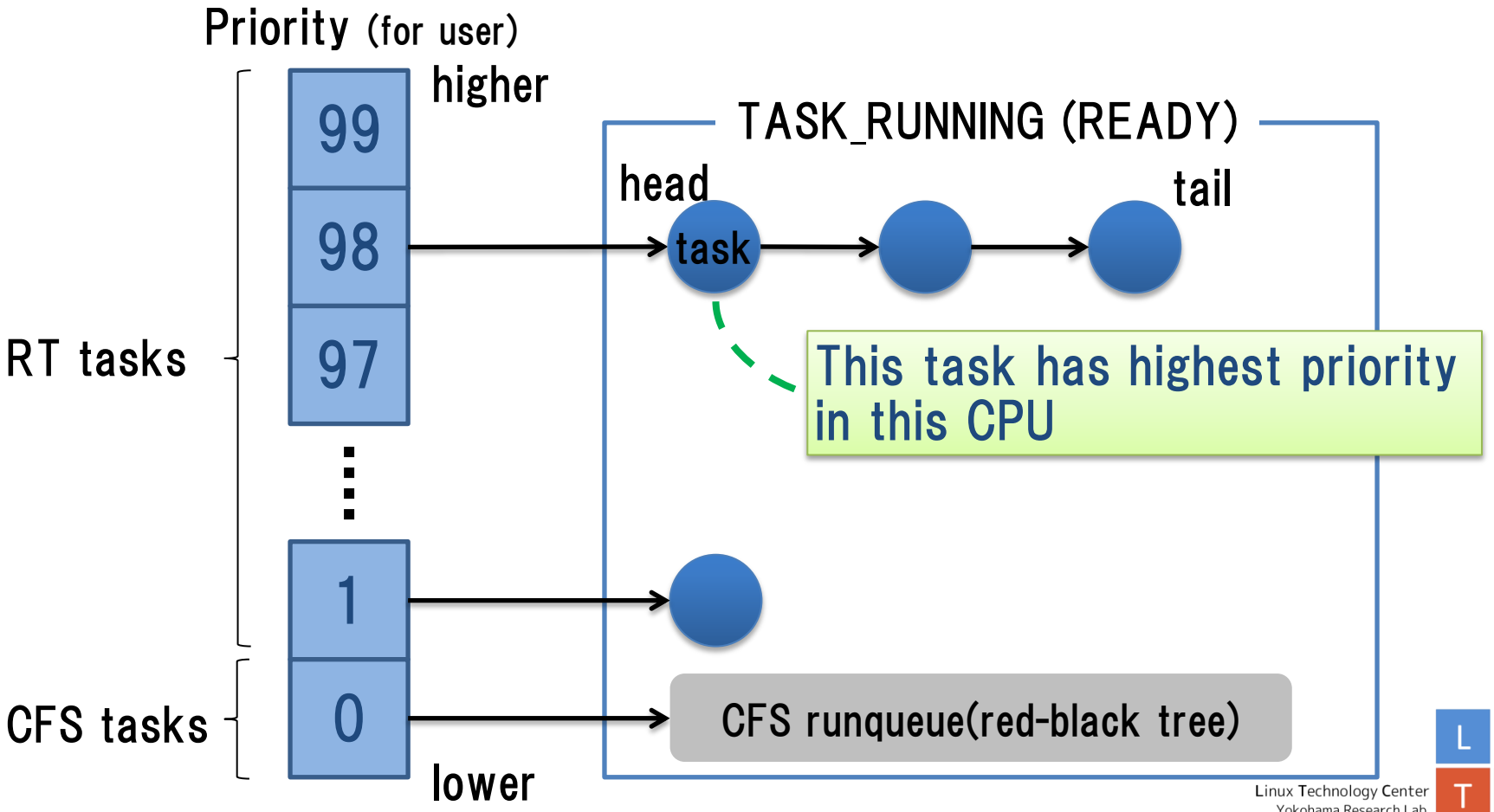
- Linux has three types of schedulers
  - Completely Fair Scheduler (CFS)
    - Policy: SCHED_OTHER(Default), _BATCH, _IDLE
    - Task has dynamic-priority based on time slice (non-deterministic)
  - Real-time scheduler
    - Policy: SCHED_FIFO, _RR
    - Task has fixed-priority (deterministic)
  - Deadline scheduler
    - Policy: SCHED_DEADLINE
    - Merged in 3.14

Linux Technology Center
Yokohama Research Lab.

Fixed-priority scheduling (cont.)

HITACHI
Inspire the Next

- There are per-cpu runqueues in Linux
  - RT tasks are always dispatched prior to CFS tasks



Linux Technology Center
Yokohama Research Lab.

Kernel preemption

HITACHI
Inspire the Next

- Without preemption support: when higher priority tasks wake up, they are delayed until current task exits from a syscall or yields explicitly

- CONFIG_PREEMPT
  - Enables forcible context-switch even in kernel mode
  - Though spin-locks disable preemption
  - Improves response latency of high priority tasks

High resolution timer（hrtimer）

HITACHI
Inspire the Next

- Some real-time systems start processing triggered by periodic timer events
  - But timer_list subsystem has only tick(jiffies) resolution
    - Tick intervals of 1~10ms are too long
    - Tick-independent software timer is necessary
- CONFIG_HIGH _RES_TIMERS
  - Enables software timer mechanism which supports nanosecond resolution
  - Userspace programs using nanosleep, itimers or posix-timers can benefit from this feature

IRQ thread

HITACHI
Inspire the Next

- Move a part of the interrupt handler code to a kernel thread
  - reduce interrupt-disabled section
  - prioritize interrupt handlers

- CONFIG_IRQ_FORCED_THREADING
  - Force threading of all interrupt handlers
    - except handlers marked IRQF_NO_THREAD
  - Enabled by boot parameter “threadirqs”

IRQ thread (cont.)

```
# ps |grep irq
    3 root       0:00 [ksoftirqd/0]
   64 root       0:00 [irq/12-edma]
   65 root       0:00 [irq/14-edma_err]
  203 root       0:00 [irq/70-omap_i2c]
  921 root       0:00 [irq/30-omap_i2c]
  956 root       0:00 [irq/18-musb-hdr]
  959 root       0:00 [irq/19-musb-hdr]
  967 root       0:00 [irq/75-rtc0]
  968 root       0:00 [irq/76-rtc0]
  977 root       0:00 [irq/64-mmc0]
  978 root       0:00 [irq/166-mmc0]
  988 root       0:00 [irq/77-wkup_m3]
  989 root       0:00 [irq/78-wkup_m3_]
  990 root       0:00 [irq/120-smartre]
  991 root       0:00 [irq/121-smartre]
 1006 root       0:00 [irq/72-OMAP UAR]
```

Others

HITACHI
Inspire the Next

- PI (Priority Inheritance) mutex
- Preemptive RCU
- RT group scheduling
- RT throttling
- Full tickless (no_hz)
- etc…

Linux Technology Center
Yokohama Research Lab.

PREEMPT_RT patchset

HITACHI
Inspire the Next

- provides more deterministic behavior
  - https://www.kernel.org/pub/linux/kernel/projects/rt/
  - It consists of lots of patches lowering maximum latency
    - There are 318 patches in 3.14.2-rt3

- CONFIG_PREEMPT_RT_FULL
  - allows “full preemption”
    - nearly all of the kernel can be preempted
    - replaces spin-locks with mutexes (Sleeping spin-lock)
    - many other improvements

Linux Technology Center
Yokohama Research Lab.

HITACHI
Inspire the Next

Tuning points for evaluation

Linux Technology Center
Yokohama Research Lab.

Tuning points for evaluation

HITACHI
Inspire the Next

- Clocksource resolution
- Task priority
- Task switching cost
- Page faults
- Multi-core
- Lock debugging

Clocksource resolution

HITACHI
Inspire the Next

- Getting and updating clock time depend on clocksource
  - gettimeofday(2), clock_gettime(2)
- This means that time measurement is affected by clocksource resolution
  - Confirmation is necessary before measurement

```
# dmesg | grep clock
[0.000000] OMAP clocksource: GPTIMER1 at 24000000 Hz
[0.000000] sched_clock: 32 bits at 24MHz, resolution 41ns,
wraps every 178956ms
```

Linux Technology Center
Yokohama Research Lab.

Task priority

HITACHI
Inspire the Next

- We need to use RT policy and priorities for deterministic scheduling
  - sched_setscheduler(2) can change those scheduling attributes of specified tasks
  - In pthread library, new thread's scheduling attributes can be set by pthread_attr_setschedparam(3)
    - Do not forget to call pthread_attr_setinheritsched(3) with PTHREAD_EXPLICIT_SCHED

Task switching

HITACHI
Inspire the Next

- Process switching cost is significantly larger than thread switching
  - Process switching needs to flush TLB
  - If your real-time application is composed of lots of processes, process switching measurement is necessary

Linux Technology Center
Yokohama Research Lab.

Page faults

HITACHI
Inspire the Next

- Memory allocation to userspace is usually only virtual address space allocation
  - Initial memory access causes page fault, and this produces more latency
  - Page-out to swap area also causes page faults

- To prevent page faults, mlockall(2) is used
  - All memory-mapped pages remain on physical memory after mlockall(2)

Multi-core

HITACHI
Inspire the Next

- In multi-core SoC, tasks can move from local core to remote cores
  - This migration causes additional latency
  - Tasks can be fixed to a specific core by cpuset cgroup

- Influence of unrelated interrupt handling
  - It’s effective to fix real-time tasks to cores which doesn’t handle unrelated IRQs
    - after setting IRQ affinity.

Linux Technology Center
Yokohama Research Lab.

Lock debugging features

HITACHI
Inspire the Next

- Lock debugging features add much more latency
  - Disabling them is effective

Linux Technology Center
Yokohama Research Lab.

HITACHI
Inspire the Next

Performance measurement

Linux Technology Center
Yokohama Research Lab.

Measured items［1/2］

HITACHI
Inspire the Next

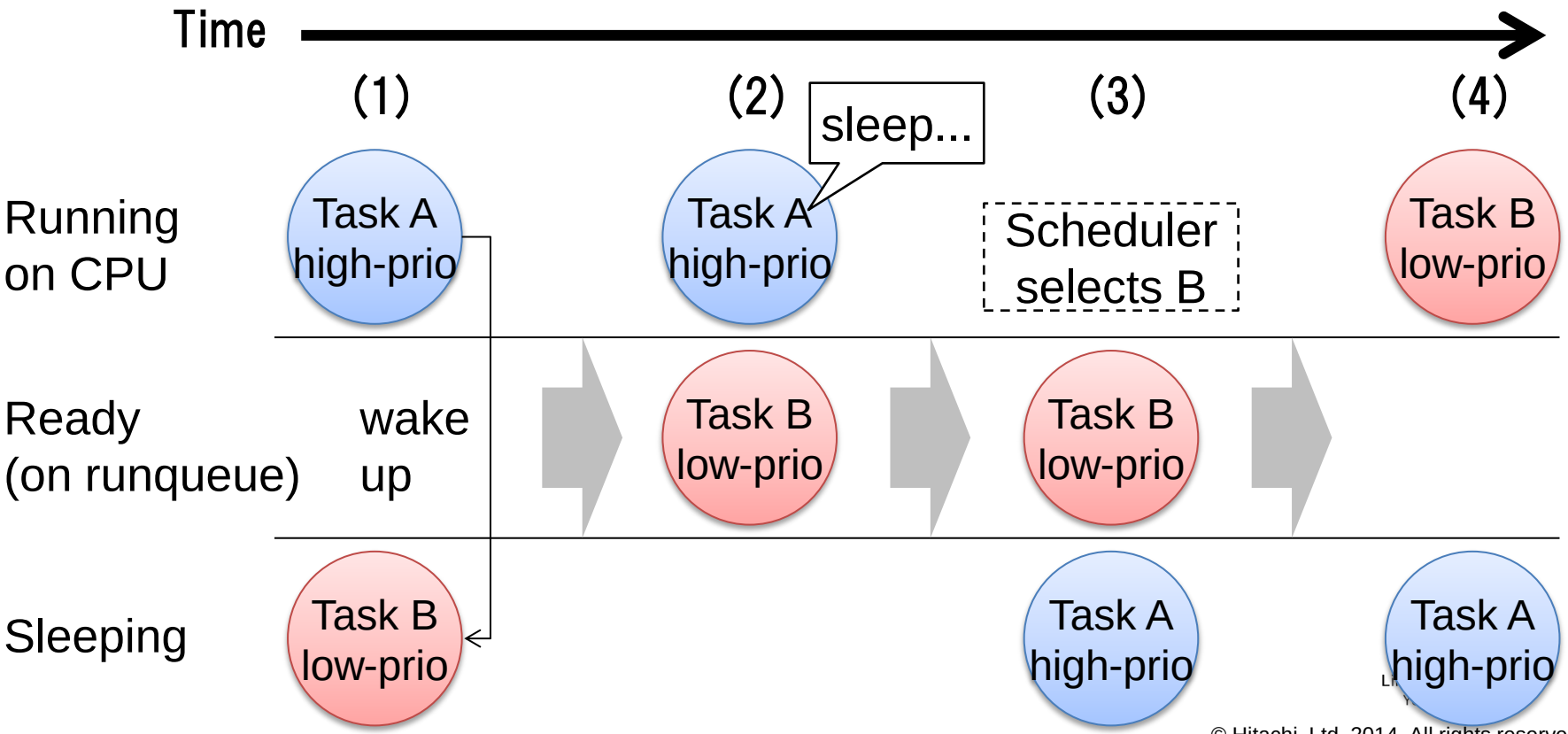
- Interrupt response time
  - Time period between expiry of a hardware timer and scheduling of a userspace task
  - measured by cyclictest (in rt-tests)
  - cyclictest -a 0 -p 99 -m -n -l 100000 -q --histogram=500
    - -a: set CPU affinity to CPU0
    - -p 99: set priority to 99
    - -m: use mlockall
    - -n: use clock_nanosleep to measure time
    - -l 100000: number of trials
    - -q: don’t print anything while testing
    - --histogram=50: print histogram 0-49us after the test

Linux Technology Center
Yokohama Research Lab.

Measured items [2/2]

HITACHI
Inspire the Next

- Task switching time
  - measured by own program (though ptsematest exists…)
    - Cause thread switching by POSIX semaphore 100k times

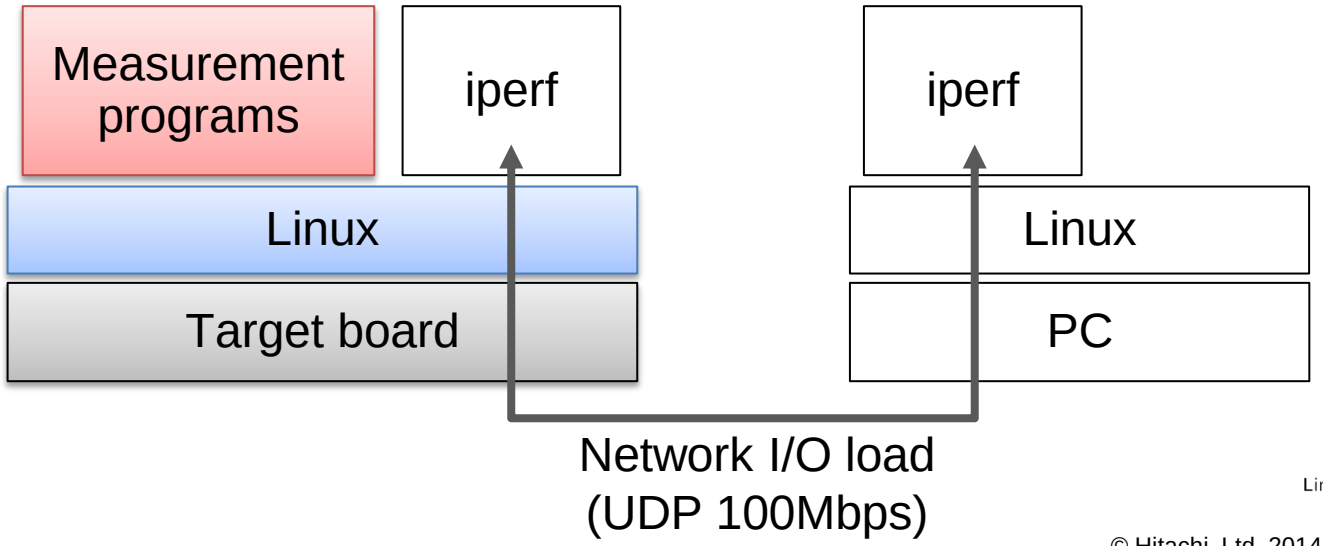


Evaluation environment

HITACHI
Inspire the Next

Board	BeagleBone Black	Altima Helio board
SoC	TI Sitara AM3359	Altera Cyclone V
CPU	ARM Cortex-A8 (1GHz)	ARM Cortex-A9 (800MHz x2)
L1 cache (inst. / data)	32KB / 32KB	32KB / 32KB
L2 cache	256KB	512KB
Main memory	512MB	1GB
Linux version	3.14.0	3.14.0 [1]
PREEMPT_RT version	3.14.0-rt1	3.14.0-rt1

[1] http://rocketboards.org/gitweb/?p=linux-socfpga.git



Linux Technology Center
Yokohama Research Lab.

Comparison

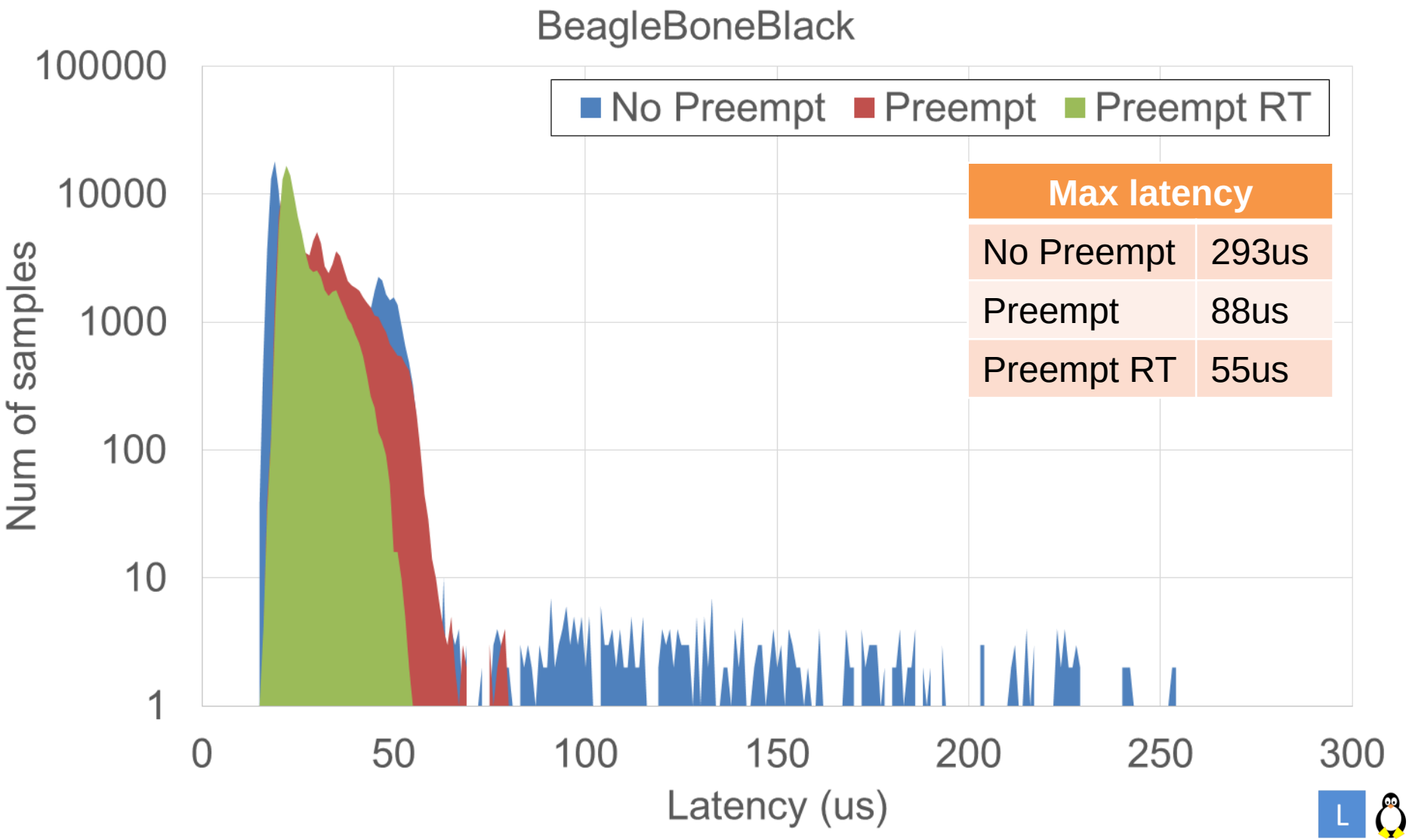
HITACHI
Inspire the Next

- Three types of kernels
  - “No preempt”
  - “Preempt”
  - “Preempt RT”

Settings	“No Preempt”	“Preempt”	“Preempt RT”
Kernel version	3.14.0	3.14.0	3.14.0-rt1
Preemption	No	**Yes**	**Yes**
Forced IRQ threading	No	**Yes**	**Yes**
Full preemption (RT patch)	No	No	**Yes**

Interrupt response time (BeagleBoneBlack)

HITACHI
Inspire the Next

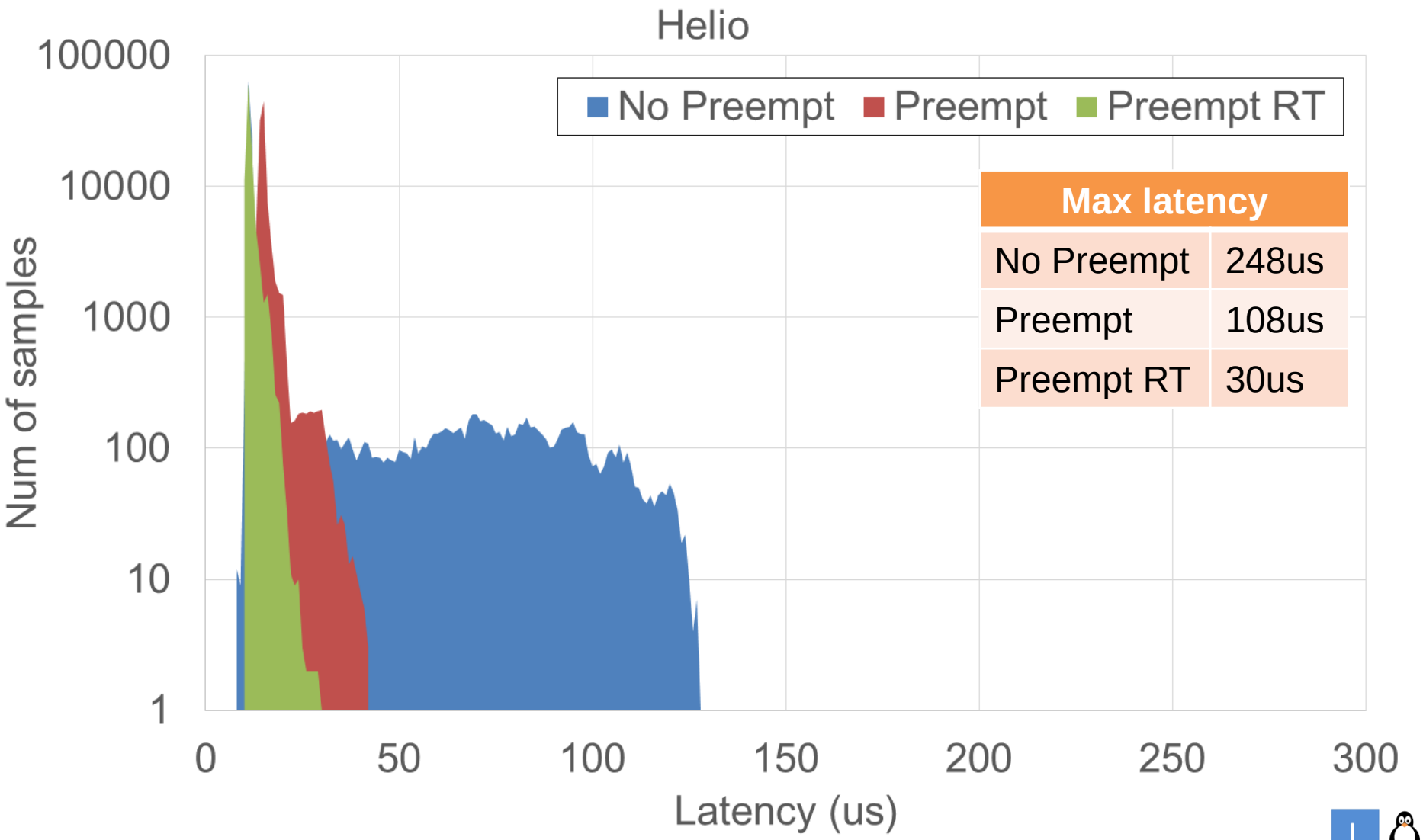


Max latency	
No Preempt	293us
Preempt	88us
Preempt RT	55us

Linux Technology Center
Yokohama Research Lab.

Interrupt response time (Helio)

HITACHI
Inspire the Next

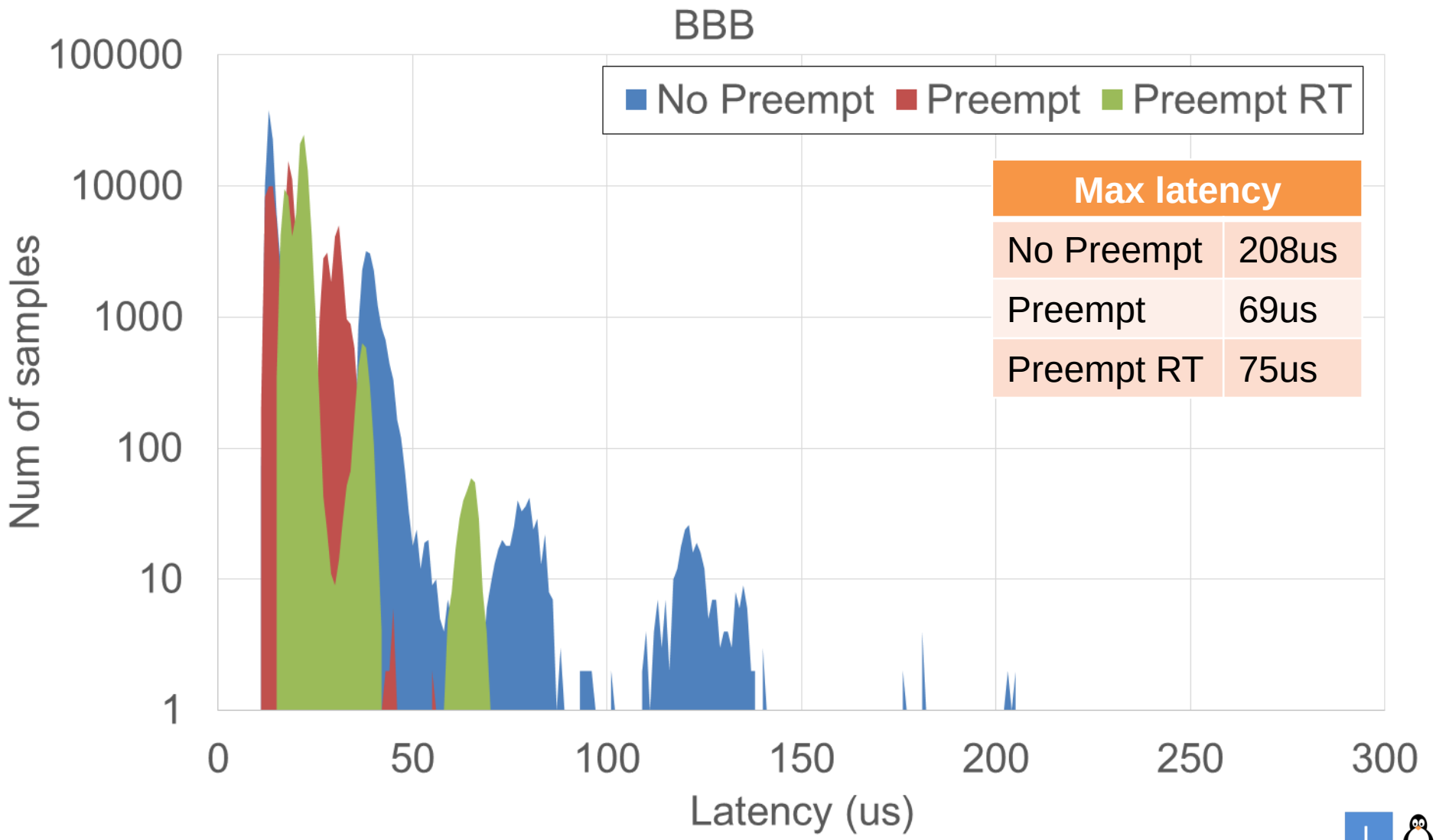


Max latency	
No Preempt	248us
Preempt	108us
Preempt RT	30us

Linux Technology Center
Yokohama Research Lab.

Task switching time (BeagleBoneBlack)

HITACHI
Inspire the Next

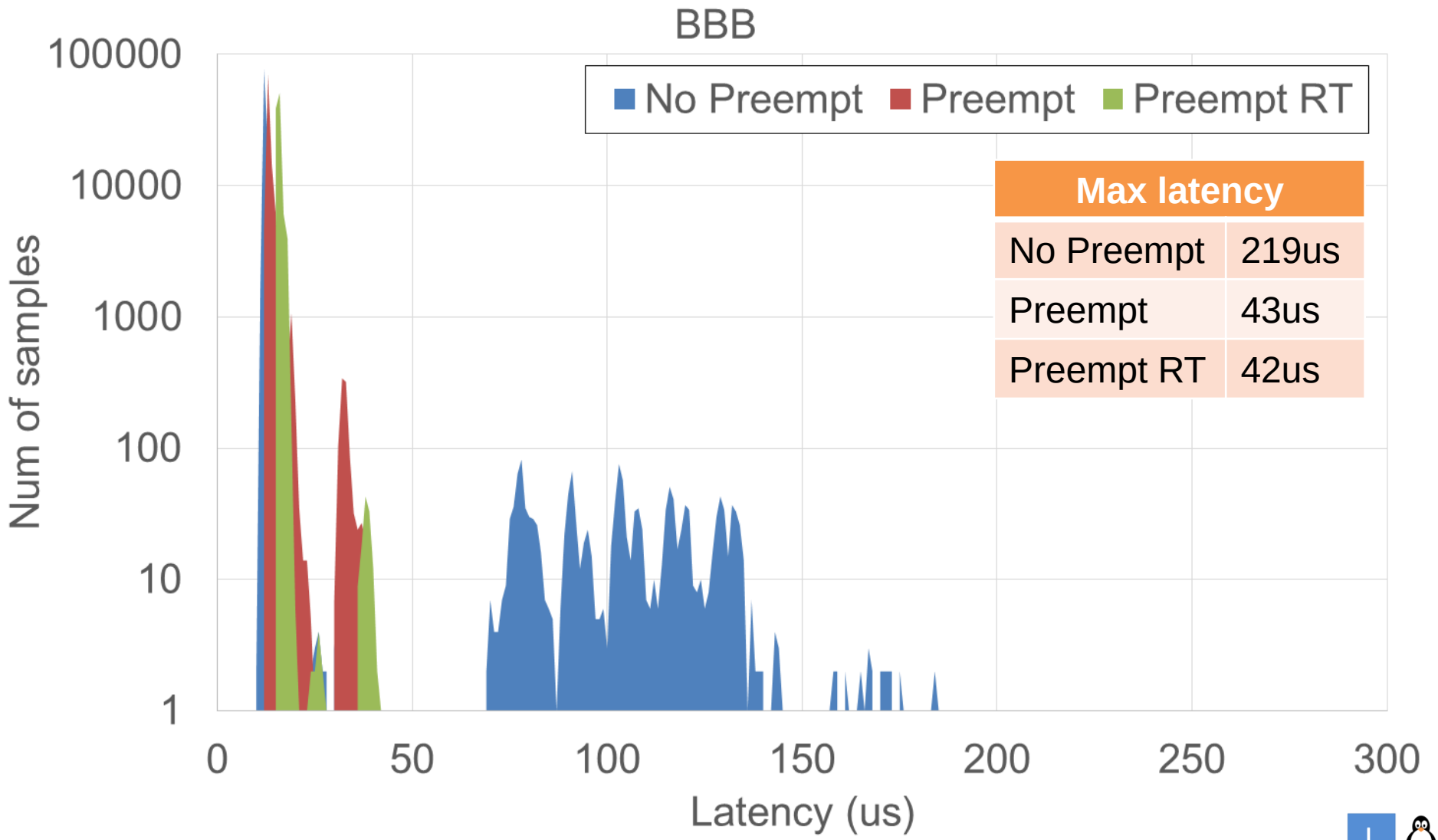


Max latency	
No Preempt	208us
Preempt	69us
Preempt RT	75us

Linux Technology Center
Yokohama Research Lab.

Task switching time (Helio)

HITACHI
Inspire the Next



Max latency	
No Preempt	219us
Preempt	43us
Preempt RT	42us

Linux Technology Center
Yokohama Research Lab.

Conclusion

HITACHI
Inspire the Next

- Current mainline Linux has lots of real-time features
- PREEMPT_RT patchset provides more deterministic behavior
- We can reduce maximum latency by using these features with appropriate configuration and tuning

Trademarks

HITACHI
Inspire the Next

- Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.

Linux Technology Center
Yokohama Research Lab.