Porting and Evaluating the Linux Realtime Preemption on Embedded Platform

Katsuya Matsubara [1)]

Hitomi Takahashi [1)]

Hisao Munakata [2)]

[1)] IGEL Co., Ltd

[2)] Renesas Solutions Corp.

Contents

igel

- Background
- Linux Preemption
- Porting PREEMPT_RT for a Renesas SuperH-based platform
- Performance Evaluation
- Conclusion

Background

igel

- Device Driver Development in Embedded World is different in the following senses:
  - Non-common new devices
    - Due to newly developed devices, it is quite hard to re-use existing device drivers
    - Tightly coupled with applications
      - Often monolithic system architecture. Application requires to manage devices directly in fine-grain.
      - Only one application dominantly uses the device.
    - Single-user, multi-task
    - IPR issue
    - Requires easiness of device driver development
      - Short development cycle

Objective

igel

- Implement device drivers in user-space
  - Easy to develop
  - Close to applications
  - keeps kernel stable
- Evaluate kernel features for user-level device driver.
  - Functionality
  - Performance
    - NPTL
    - O(1) scheduler
    - PREEMPT_RT
  - API

Preemption in Linux

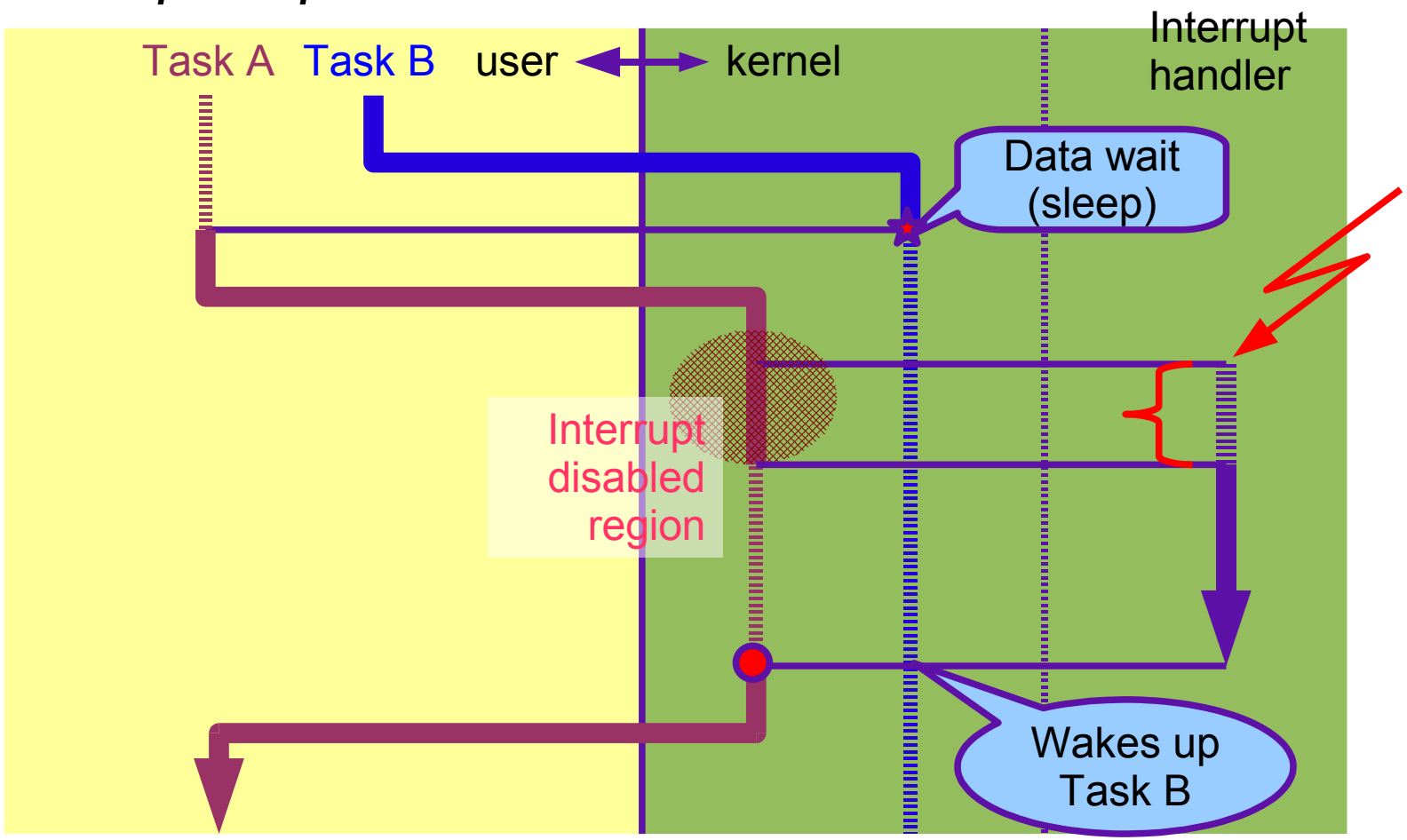
igel

- Preemption points in 2.4/2.6 PREEMPT_NONE kernel
  - At return from interrupt handling
  - At return from system call
  - When task sleeps voluntarily
- Any tasks in kernel mode cannot be preempted!
  - Tasks are interrupted only when hardware interrupts had occurred

CONFIG_PREEMPT_NONE

igel

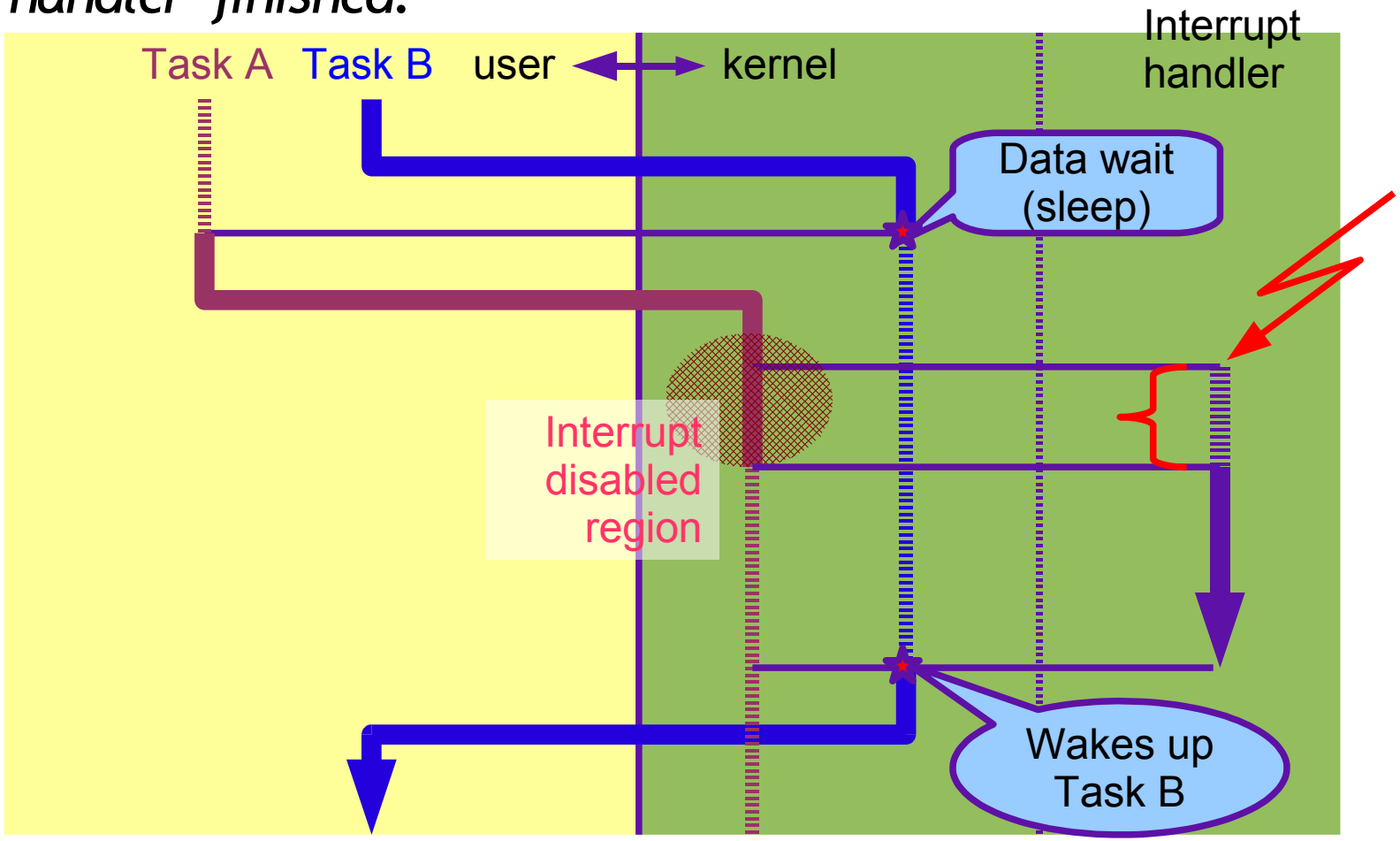
- *Never preempt tasks in kernel*



CONFIG_PREEMPT(_DESKTOP)

igel

- *The scheduler selects an appropriate task when interrupt handler finished.*

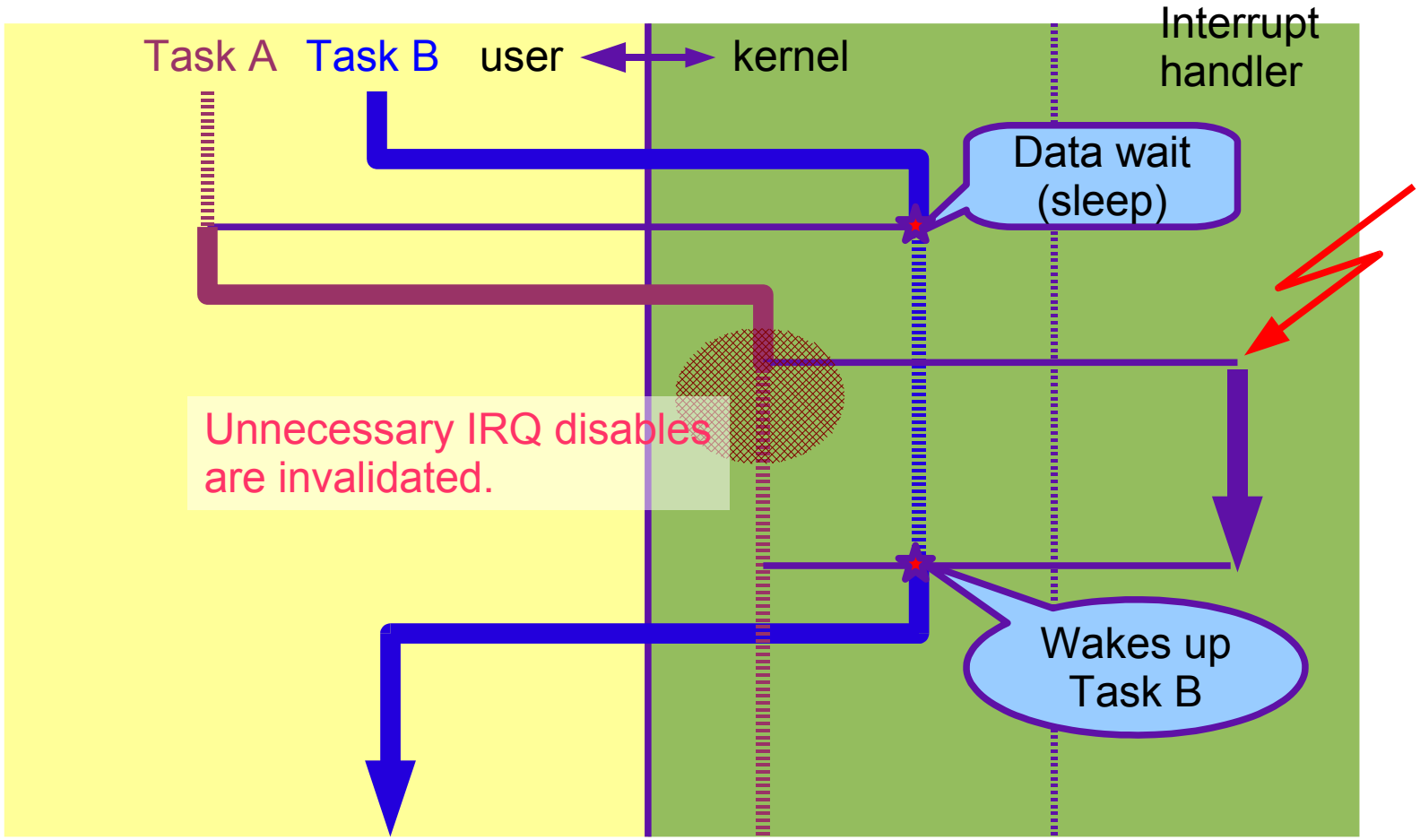


CONFIG_PREEMPT_RT

igel

- *Distributed by Ingo Molnar and a small group of core developers.*
- *Allows almost all of the kernel to be preempted.*

CONFIG_PREEMPT_RT (contd.)



Features added/modified by PREEMPT_RT

igel

- Preemption in critical regions
- Preemption in interrupt handlers
- Preemption in atomic (interrupt-disabled) operations
- (Priority inheritance of spinlocks and mutexes)
  - Merged to the mainline
- Miscellaneous optimizations
  - Cleaned up code
  - Fixed race conditions and locking problems

Preemption in critical regions

igel

- Task can be preempted even if it is in a critical region with holding a lock.
  - spinlock_t and rwlock_t have been modified to allow preemption in protected region.
  - spin_lock_irq*(), e.g. spin_lock_irqsave(), does not actually inhibit hardware interrupt.
  - Original spinlocks are kept usable with raw_spinlock_t.

Preemption in interrupt handlers

igel

- Interrupt handlers can be preempted;
  - Interrupt handlers run in process context (rather than in interrupt context)
- Some interrupt handlers marked with IRQF_NODELAY runs in interrupt context
  - e.g. CPU timer, FPU, etc.

Preemption in atomic operations

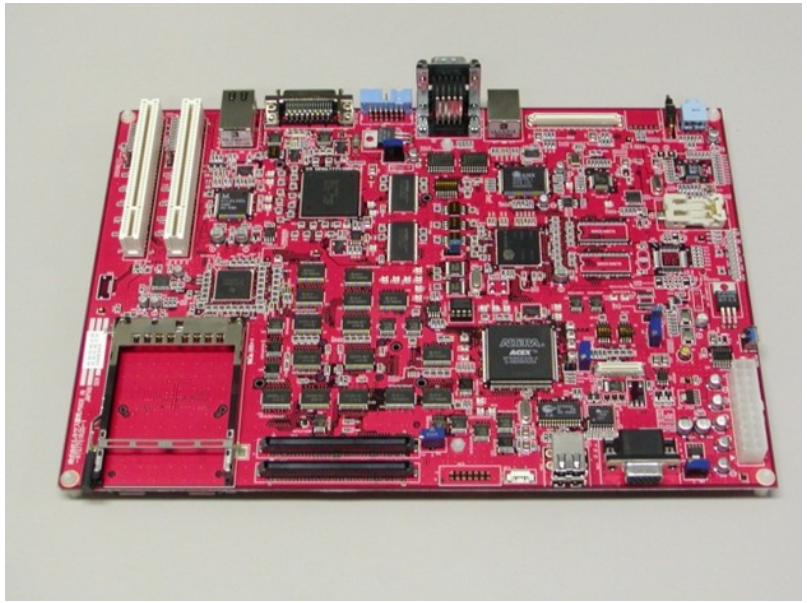
igel

- Tasks are preempted even they request interrupt disabling.
  - local_irq_save() does not actually inhibit hardware interrupt.
  - raw_local_irq_save/restore() perform with original behavior.

Porting PREEMPT_RT to a new architecture

igel

- Renesas RTS7751R2D board
  - SH7751R(SH-4 architecture)
  - Peripherals
    - Network (RTL-8139DL)
    - Serials (SCIF, SM501 UART)
    - USB (provided by SM501)
    - 2D graphics (SM501)
    - PCI buses
    - CF/PCMCIA slots
- Base software
  - Linux-2.6.21-rc5
  - patch-2.6.21-rc5-rt12



How to port PREEMPT_RT to a new architecture

igel

- Read and modify codes in arch/, include/asm-arch, and drivers/
  - Rename semaphore definitions
  - Modify context switching
  - Do refactoring of critical regions
  - Do refactoring of interrupt handlers
- Other architecture's implementations (ARM, MIPS, PowerPC, and i386) are good resourses to find out which codes should be modified.

Semaphore definitions

igel

- PREEMPT_RT provides spinlock-based semaphores and rwlocks for the priority inheritance feature.
- Original (architecture-dependent) semaphore and rwlock should be re-named to compat_*.

```
- static inline void down(struct semaphore * sem)
+ static inline void compat_down(struct compat_semaphore * sem)
 {
        might_sleep();
        if (atomic_dec_return(&sem->count) < 0)
-               __down(sem);
+               __compat_down(sem);
 }
```

Context Switching

igel

- Original schedule() is renamed to __schedule() and new schedule() is defined.
- Append TIF_NEED_RESCHED_DELAYED flag in thread information flags
  - Indicates 'reschedule on return to userspace'

```
-       tst    #_TIF_NEED_RESCHED, r0
+       tst    #_TIF_NEED_RESCHED | _TIF_NEED_RESCHED_DELAYED, r0

        .align  2
- 1:    .long   schedule
+ 1:    .long   __schedule
```

Critical Regions and Atomic Operations

igel

- Review all critical regions and atomic operations to make sure that there are no race conditions or locking problems.
  - Do refactoing the code or replace spinlocks or local_irq_save by raw_* if needed

```
-       local_irq_save(flags);
+        raw_local_irq_save(flags);
        ret = v->counter;
        if (likely(ret == old))
                v->counter = new;
-       local_irq_restore(flags);
+        raw_local_irq_restore(flags);
```

Interrupt Handlers

igel

- IRQF_NODELAY
  - This flag is automatically set for the timer with the PREEMPT_RT patch.
  - Review other handlers, and mark appropriately if they need to run in interrupt context. In our work, there was no need to mark additionally.

```
-#define IRQF_TIMER          0x00000200
+#define __IRQF_TIMER        0x00000200
 #define IRQF_PERCPU         0x00000400
 #define IRQF_NOBALANCING    0x00000800
+#define IRQF_NODELAY        0x00001000
+#define IRQF_TIMER          (__IRQF_TIMER | IRQF_NODELAY)
```

Other Changes

igel

- The Ingo's patch consists of pieces of codes which assumes that CONFIG_GENERIC_TIME is set to 'y'.
  - The patch overrides the original code, and it relies on codes that are enabled only when CONFIG_GENERIC_TIME is set.
    - In SH, this was the case.
  - Fixed the codes to use original codes if CONFIG_GENERIC_TIME is not set.
- Replaced a spinlock defined at 8250 serial driver
  - SCIF, which is SH internal serial I/O, driver uses spinlock_irq_* for atomic and mutual exclusion.
  - Replaced the spinlock with raw_spinlock.

Diff from patch-2.6.21-rc5-rt12 for SH platform (1/3)


```
arch/sh/kernel/cpu/clock.c            |    2
arch/sh/kernel/cpu/sh4/sq.c           |    2
arch/sh/kernel/entry-common.S         |    8 +--
arch/sh/kernel/irq.c                  |    2
arch/sh/kernel/process.c              |    8 ++-
arch/sh/kernel/semaphore.c            |   14 ++++--
arch/sh/kernel/sh_ksyms.c             |    9 +---
arch/sh/kernel/signal.c               |    7 +++
arch/sh/kernel/time.c                 |    2
arch/sh/kernel/traps.c                |    2
arch/sh/mm/cache-sh4.c                |   12 ++---
arch/sh/mm/init.c                     |    2
arch/sh/mm/pg-sh4.c                   |    8 +--
```

Diff from patch-2.6.21-rc5-rt12 for SH platform (2/3)


```
arch/sh/mm/tlb-flush.c                  |   20 ++++-----
arch/sh/mm/tlb-sh4.c                    |    4 -
include/asm-sh/atomic-irq.h             |   24 +++++------
include/asm-sh/atomic.h                 |    8 +--
include/asm-sh/bitops.h                 |   24 +++++------
include/asm-sh/pgalloc.h                |    2
include/asm-sh/rwsem.h                  |   46 ++++++++++++----------
include/asm-sh/semaphore-helper.h       |    8 +--
include/asm-sh/semaphore.h              |   61 ++++++++++++++++++++-----------
include/asm-sh/system.h                 |   12 ++---
include/asm-sh/thread_info.h            |    2
```

Diff from patch-2.6.21-rc5-rt12 for SH platform (3/3)


```
include/linux/serial_core.h          |    2
kernel/hrtimer.c                     |    4 +
kernel/time.c                        |    4 +
kernel/time/ntp.c                    |    4 +
kernel/timer.c                       |    6 ++
```

29 files changed,
178 insertions(+), 131 deletions(-)

Performance Evaluation: Impact to ULDD

igel

- Proposed ULDD framework at ELC2006
  - http://www.igel.co.jp/file/uldd060411celfelc2006.pdf
- Userspace I/O driver (UIO)

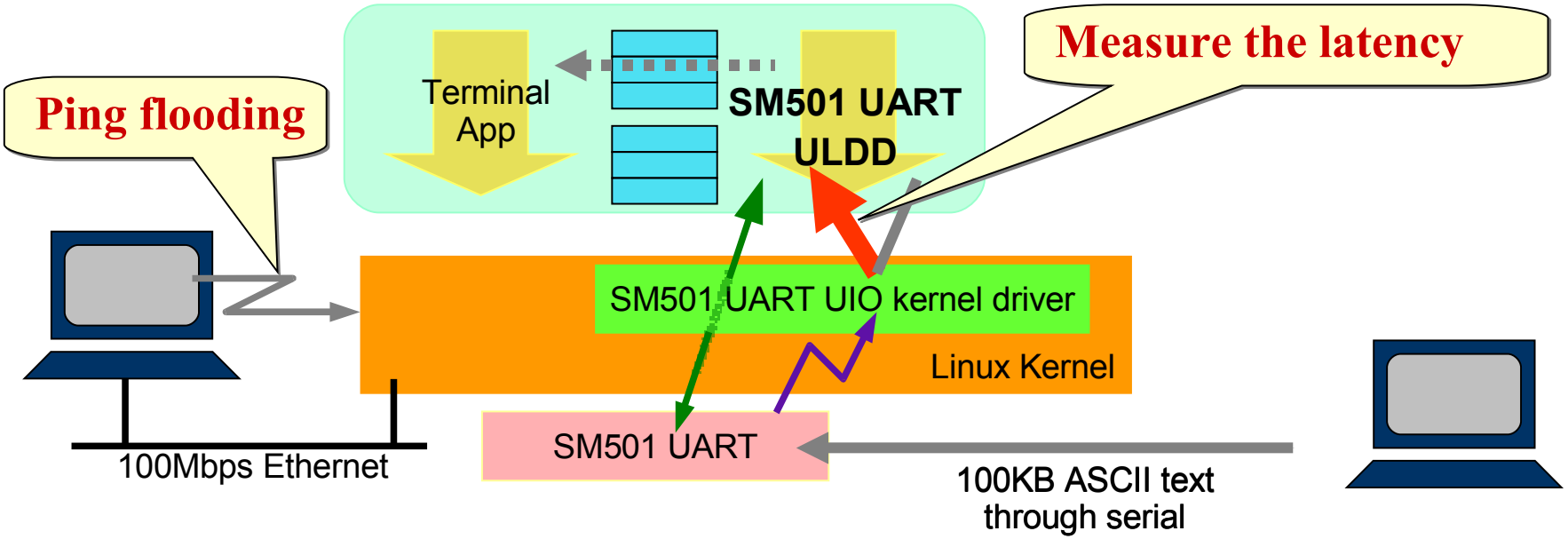
  From: Hans J. Koch

  This interface allows the ability to write the majority of a driver in userspace with only very small shell of a driver in the kernel itself. It uses a char device and sysfs to interact with a userspace process to process interrupts and control memory accesses.

  (Quoted from Greg Kroah-Hartman's log)

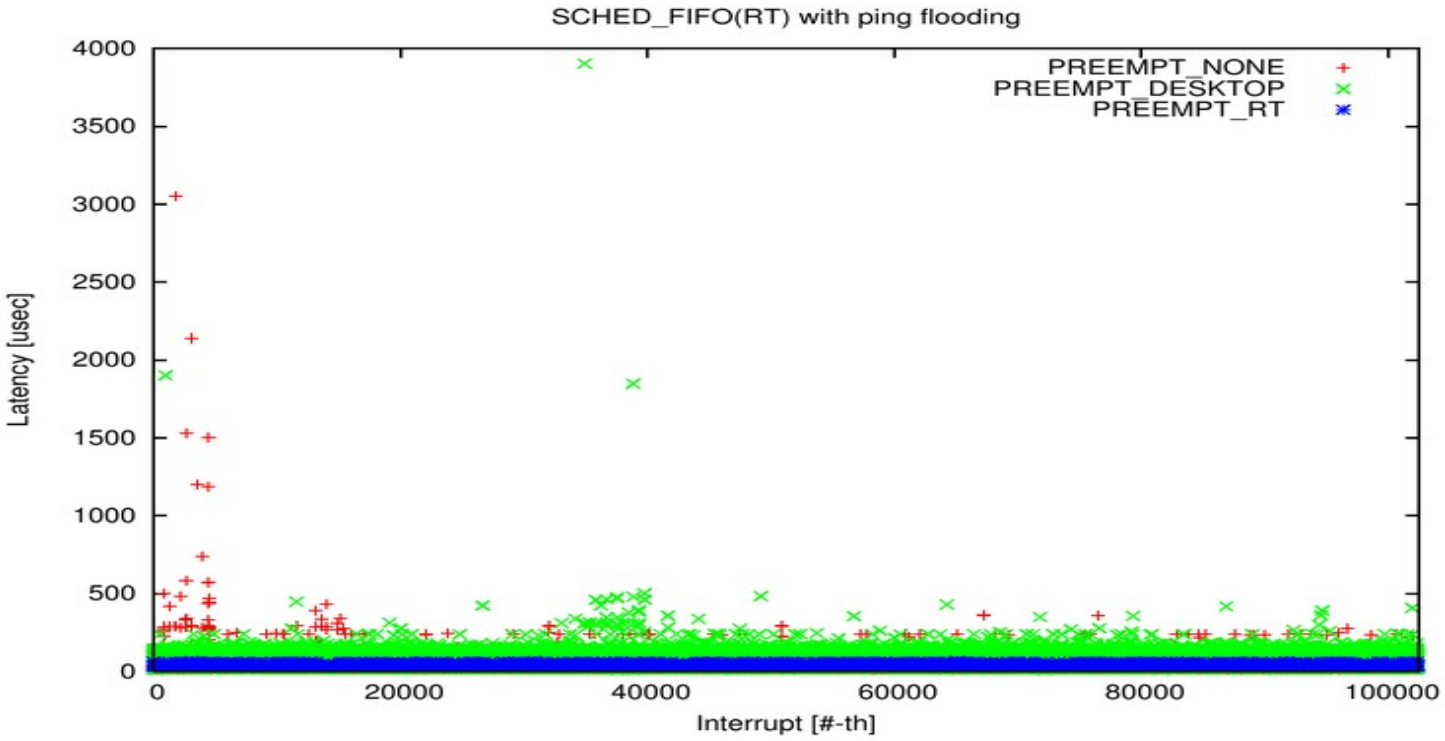
An Experiment with ULDD

- Implement a SM501 UART driver with UIO
- Measure the schedule latency (when kernel handler invokes user handler)



Result

The effectiveness of PREEMPT_RT can be particularly observed when the scheduler policy for ULDD is set to SCHED_FIFO under high system load.



Conclusion

igel

- Ported PREEMPT_RT for Renesas SH-4 architecture:
  - Modified some code in arch/sh and include/asm-sh
  - Read code of related devices and modified some code in drivers
- Implemented a user-level device driver with UIO:
  - It performs well even it is located in user space.
  - PREEMPT_RT gives a good effect on user-level drivers.

Future Work

- Submit the patch to lkml and/or linux-rt ml SOON!
- Catch up the latest version (2.6.21-rc6-rt0).
- Support other SuperH based platforms.
  - CPU architecture families; SH-3, SH-4A
  - Peripheral devices on the other SH platforms

References

- PREEMPT_RT
  - patches
    - http://people.redhat.com/mingo/realtime-preempt/
  - RT Wiki
    - http://rt.wiki.kernel.org/index.php/Main_Page
- UIO
  - http://www.kroah.com/log/linux/uio.html