

# **Building IVI Product on ARM 64 bit architecture**

**2016/7/13**

**Panasonic Industrial Devices Systems and Technology Co., Ltd**

**Yohei Ikeuchi**

# Who am I

- ◆ From embedded OS distribution provider company PIDST, a member of the Panasonic Group
- ◆ We have been supporting consumer customers of Panasonic semiconductor during over 15 years
  - Scope : Linux (kernel 2.6.11 ~) , Real-Time OS (uITRON 4.0) , Toolchain (GCC, LLVM/Clang)
  - Product area : Mobile, Blu-ray, Digital TV, Camera
- ◆ We also have been supporting automotive customer of various ARM® SoC in recent 3 years

# Agenda

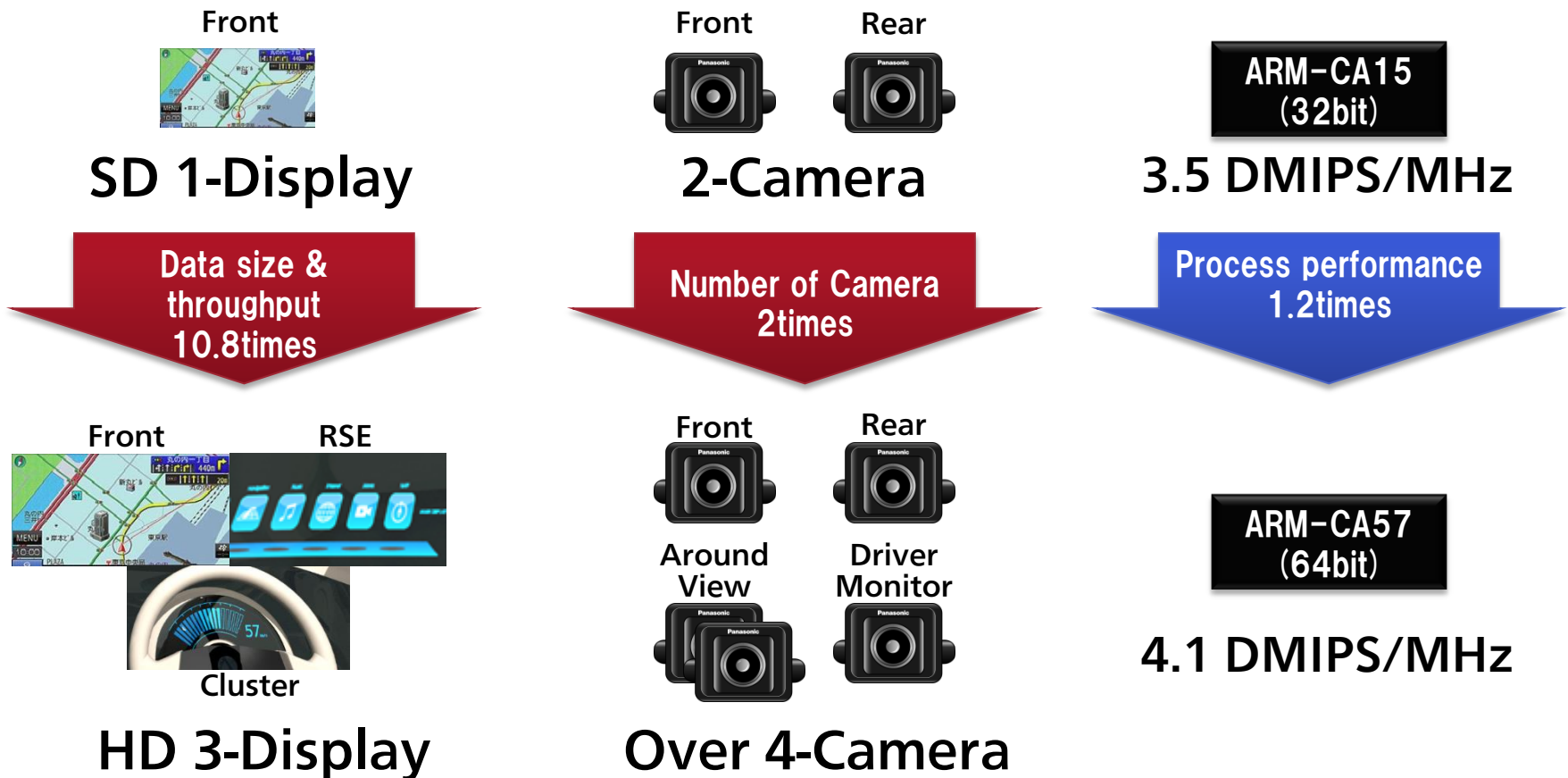
1. Why 64bit Linux is needed?
2. OSS/Application introduction to AArch64 environment
3. Response performance of AArch64 interruption

1. **Why 64bit Linux is needed?**
2. OSS/Application introduction to AArch64 environment
3. Response performance of AArch64 interruption

# IVI system has become more advanced system

## ◆ Evolution of the IVI system does not stop

- Functional expansion require more process performance & memory
- Increase of connected devices require more quick response



# The difference between 32bit Linux and 64bit Linux

## ◆ Considering the enlargement of the system, Adoption of 64bit Linux is necessarily

- Increase the memory that Kernel can be used  
(for kernel driver, CMA : Continuous Memory Allocator)
- Increase of Stack size

## ◆ Memory management

LPAE : Large Physical Address Extension

|                  |                | AArch32 | AArch32 (LPAE *)    | AArch64                                  |
|------------------|----------------|---------|---------------------|--|
| Physical address | Address widths | 32bit   | 32bit, 36bit, 40bit | 32bit, 36bit, 40bit, 42bit, 44bit, 48bit |
|                  | Valid size     | 4GB     | 4GB, 64GB, 1TB      | 4GB, 64GB, 1TB, 4TB, 16TB, 256TB         |
| Logical address  | Address widths | 32bit   | 32bit               | 39bit, 42bit, 48bit,                     |
|                  | Valid size     | 4GB     | 4GB                 | 512GB, 4TB, 256TB                        |
| Page size        |                | 4KB     | 4KB                 | 4KB or 64kB                              |

## ◆ Stack size

| AArch32        | AArch64    |
|----------------|------------|
| 8kB (SVC Mode) | 16KB (EL1) |


1. Why 64bit Linux is needed?
- 2. OSS/Application introduction to AArch64 environment**
3. Response performance of AArch64 interruption

# Can we build OSS for ARM 64-bit?

## ◆ ARM 64-bit Build environment

Introduction status of the Yocto ARM 64-bit build for OSS 106-module that Panasonic had been utilized in ARM 32bit.

|                  | Yocto Fido 1.8.0<br>(2015.9) | Yocto Krogoth 2.1<br>(2016.3) |
|------------------|------------------------------|-------------------------------|
| OSS in Yocto     | 37                           | 48                            |
| OSS out of Yocto | 69                           | 57                            |

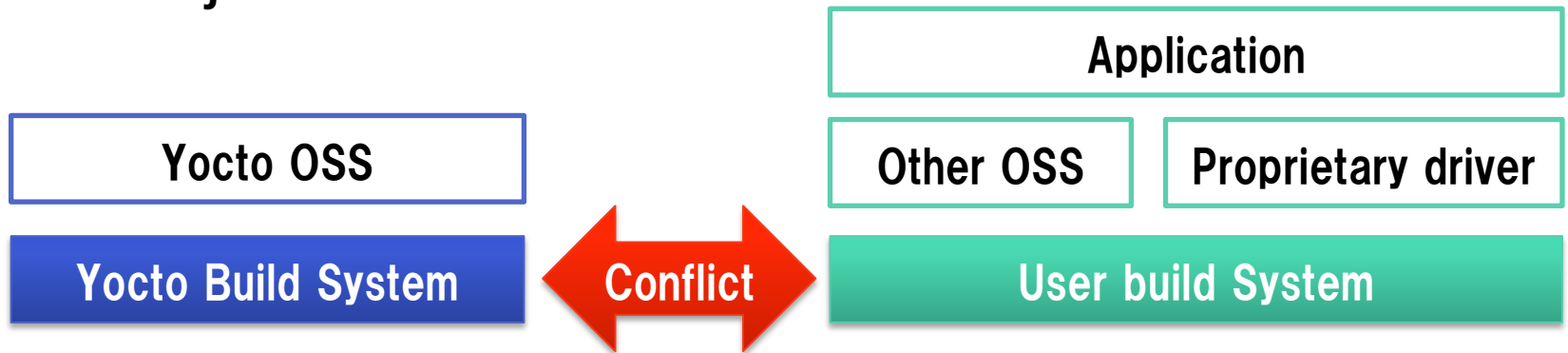


- OSS modules that can be Build in the ARM 64-bit and added to the Yocto increased almost 30% in this half a year (+11 modules)
- Many of the OSS project has been devised for the ARM 64-bit Build with their own ideas, but we are released from the problem of dependency between other OSS by the introduction of OSS as part of Yocto
- Yocto is easy to setup, because build tools and OSS modules are provided in the set



# Is Yocto enough for the product?

## ◆ Yocto is just a baseline



- ◆ All of the OSS that necessary to IVI Product is not satisfied with only the Yocto
  - OSS in the Yocto can build in the Tool Chain of the same version of the Yocto Build System
  - However, for the 64-bit build other OSS, it is often needed a newer version of the Tool Chain
- ◆ Not only OSS, Proprietary driver and Application is essential in terms of diversion of existing assets
- ◆ Conflict is often between Yocto Build System and Conventional Build system
- ◆ It is often for Proprietary driver and Application, that can not build in 64-bit environment and can build but it does not work

## ◆ Example of conflict between OSS and toolchain

- Install to “/lib64”
  - If there is no explicit specification, the library is normally install to “/lib”
- Error in the assembler code (openssl)
  - Because the assembler does not support AArch64 , build with the option that does not use the assembler, as “no-asm”
- “configure” does not work properly
  - In some cases fixed by “autoreconf”  
Ref : <https://wiki.linaro.org/RikuVoipio/TemplateRequest>
  - However, there are also cases that “autoreconf” causes failure (libsvg)  
Ref : <https://sourceforge.net/p/gtkextra/bugs/96/>
  - In order to operate correctly “configure”, it is essential to correctly set the “prefix”, “exec-prefix”, “libdir”
    - ✓ It is dangerous to set an absolute path name, because refer the path of the host-linux to run the cross-build.
    - ✓ Especially if wrong library is linked, can't detect the error until execution
  - “configure --help” is sometimes wrong (libgcrypt-1.6.5)

**Q : Can we execute application for 32-bit on ARM 64-bit environment, if re-compile for ARM 64-bit?**

**A : Basically yes, but incorrect source codes are No.**

◆ **AArch64 supports 32-bit application**

- **Supports ARMv7 Linux EABI**
- **Supports both ARM and Thumb-2 32bit user tasks**
- **Address space limited to 4GB**

◆ **However, the application that does not have consideration of the following does not work**

- **Incorrect length of the data type**
  - **Length of “long” and “pointer”**
  - **Implicit sign extension**
- **Rewriting of inline assembler**
  - **Instruction set is different**
  - **Register specifications and size are also different**

## ◆ Bit length of the data type

|           | AArch32 (A32/ILP32) | AArch64 (A64/LP64) | LLP64 |
|-----------|---------------------|--------------------|-------|
| char      | 8                   | 8                  | 8     |
| short     | 16                  | 16                 | 16    |
| int       | 32                  | 32                 | 32    |
| long      | 32                  | 64                 | 32    |
| long long | 64                  | 64                 | 64    |
| float     | 32                  | 32                 | 32    |
| double    | 64                  | 64                 | 64    |
| void*     | 32                  | 64                 | 64    |

## ◆ Bit length of “int” and “long” are different

- Implicit type conversion between “int”, “long” and “pointer” is not work correctly  
Ex) “long” variable posted to printf () as “%d” is not displayed correctly
- There are many cases that int used without consideration, but misalignment occurs when intentionally “int” is used
- However, heavy use of “long” causes enlargement of the data, proper use is important
- Even as it was 64-bit application, there is a possibility that the same problem occurs in the LLP64 application (Win64 application)

## ◆ Size of Structure

### ■ Sample code

```
struct bar {  
    int    i;  
    long   j;  
    int    k;  
    char*  p;  
};  
:  
printf ( "sizeof ( struct bar ) = %d\\n" , (int) sizeof (struct bar) );
```

### ■ Execution result

|         |                            |
|---------|----------------------------|
| AArch32 | sizeof ( struct bar ) = 16 |
| AArch64 | sizeof ( struct bar ) = 32 |

## ◆ Size of Union

### ■ Sample code

```
typedef union {  
    double  d;  
    long    l[2];  
} long_union_t;  
:  
printf ( "sizeof (long_union_t ) = %d\\n" , (int) sizeof (long_union_t) );
```

### ■ Execution result

|         |                              |
|---------|------------------------------|
| AArch32 | sizeof ( long_union_t ) = 8  |
| AArch64 | sizeof ( long_union_t ) = 16 |

- Not only the size of individual variables, also size of padding by alignment is changed
- serious problems does not occur in a general code, but there is a possibility that the problem occurs in the case of embedded code that include defines of fixed size for memory reduction

## ◆ Size of Pointer

### ■ Incorrect code

```
int a [N];  
int *p;  
int addr;  
:  
addr = (int) a;  
p = (int*) addr;
```

By casting a pointer type to int type, high-order bit is lost.

### ■ Correct code

```
#include <stdint.h>  
  
int a [N];  
int *p;  
intptr_t addr;  
:  
addr = (intptr_t) a;  
p = (int*) addr;
```

Should use the “intptr\_t” that can hold the address value.

### ■ Incorrect code

```
void *p, *q;  
printf (“num = %d¥n”, q-p);
```

Can't get the correct display.

### ■ Correct code

```
void *p, *q;  
printf (“num = %ld¥n”, q-p);
```

Calculation results of the pointer is “long”.  
It is often case that the result of the subtraction between addresses is used as int.

- Incorrect cast of the pointer is alerted by the GCC
- It is fatal in the 64-bit environment if ignore the warning for pointer casting

## ◆ Magic number

### ■ Incidental sample of Magic number

|            |  |
|------------|--|
| 4          | Byte number of pointer                   |
| 32         | Bit number of pointer                    |
| 0x7FFFFFFF | Signed integer maximum value, Mask value |
| 0x80000000 | Signed integer minimum value, Mask value |
| 0xFFFFFFFF | The maximum value of the variable, -1    |

- If there is even one common mistake, unexpected behavior occurs

## ◆ Definition of “time\_t”

```
typedef long int _time_t
```

```
( /usr/src/eglibc/eglibc-linaro-2.19-2014.04/bits/types.h:139 )
```

- The length of the “time\_t” is 64-bit in AArch64
- Based on the problem of year 2037, software that has received time information as “int” should be modified to “time\_t”
- “time\_t” has been used “struct timeval” and “struct timepsec”
- On the other hand, in the case of recording the data of “struct timeval” to file, In order to avoid a change in the file format, it should be utilized 32-bit format intentionally (ex : ctxget)

- ◆ The primary detection of dangerous code
  - Delete of the GCC Warnings is the precondition
  - Incorrect code can be partly detected by static analysis
  - It is also effective static analysis by the OSS
    - Splint (Static analysis tool)
      - ✓ GPL–licensed
      - ✓ Possible to detect the basic danger codes such as Null Dereferences and Buffer overflow
      - ✓ In order to obtain the necessary warning only, suppression of the message is possible by the annotation
      - ✓ Detect the cast to “int” from “long”

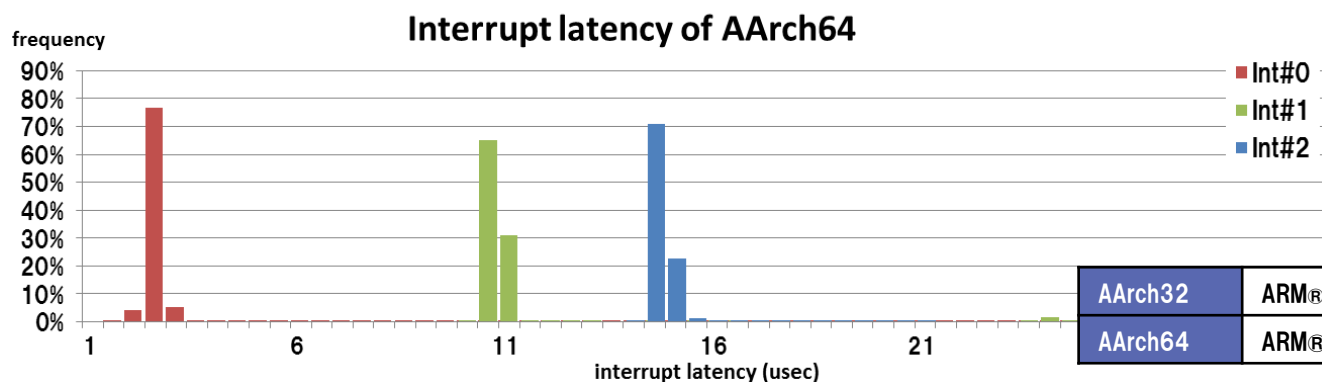
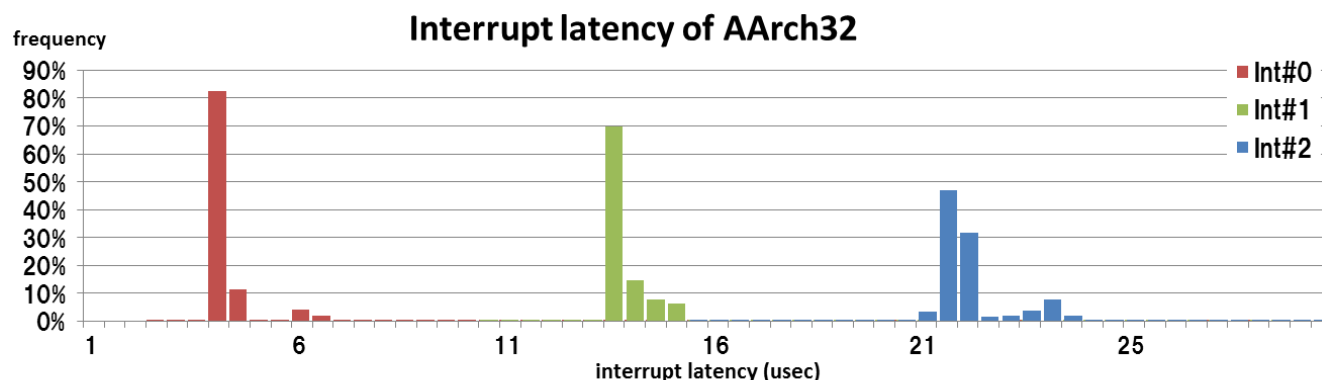


1. Why 64bit Linux is needed?
2. OSS/Application introduction to AArch64 environment
- 3. Response performance of AArch64 interruption**

# Response performance of interruption : AArch32 vs AArch64

## ◆ Interrupt latency AArch32 vs AArch64

- Measure the latency of up to IRQ vector beginning from the interrupt generation in a state of the forced three-stage multiple interrupts
- Typical latencies are improved by 30% on average  
(It is more improved against 17% which is the difference of DMIPS value)
- processing time of interrupt handler is after the latency, so the 2nd and 3rd interrupt appears to more delay
- Low latency of 3rd interrupt probably be caused by the cache hit

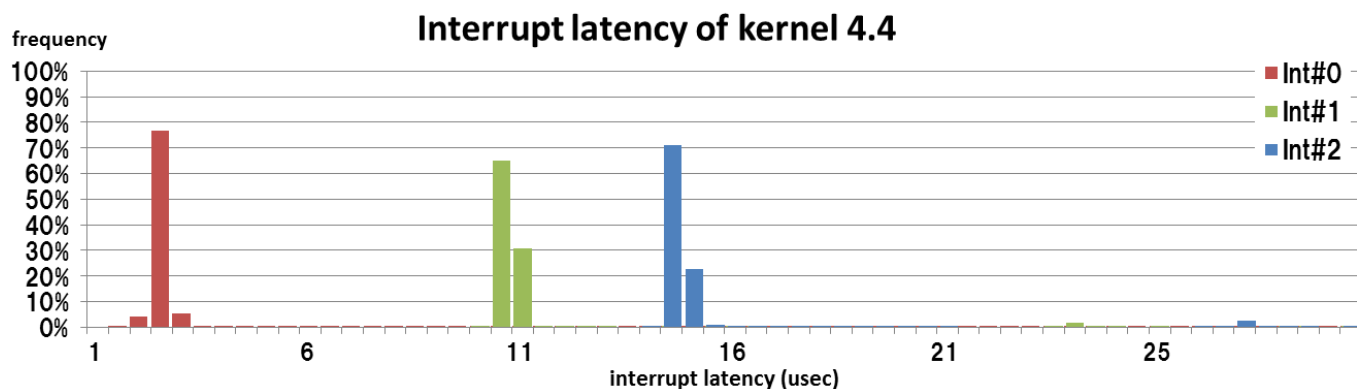
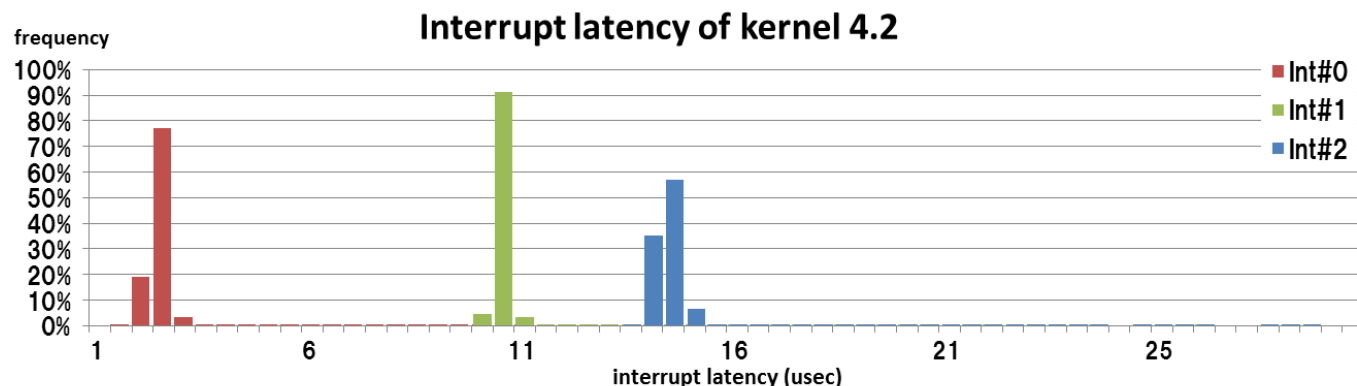


|         |                              |
|---------|------------------------------|
| AArch32 | ARM® Cortex-A15 1.5GHz 2Core |
| AArch64 | ARM® Cortex-A57 1.5GHz 4Core |

# Response performance of interruption : kernel 4.2 vs 4.4

## ◆ Interrupt latency kernel 4.2 vs kernel 4.4

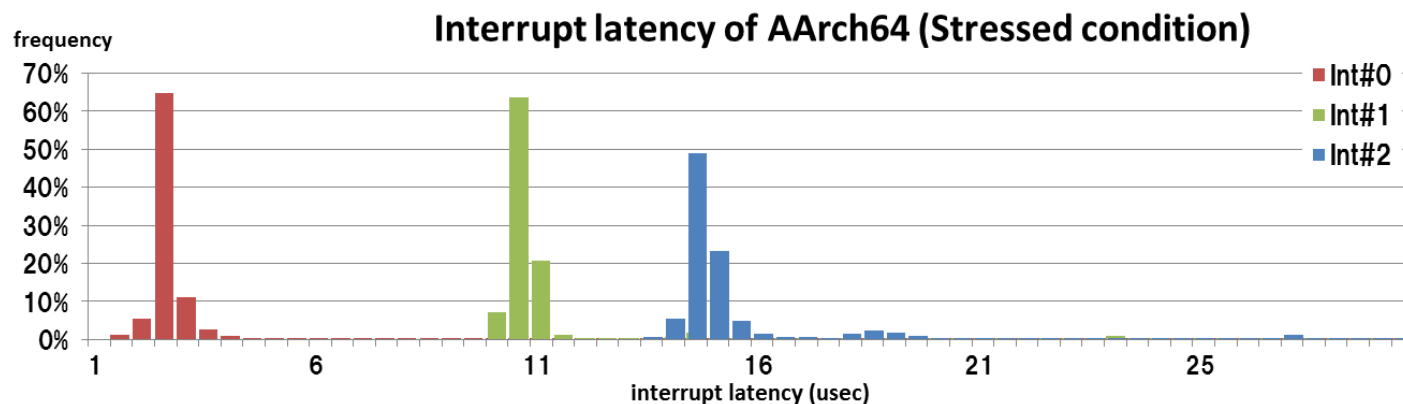
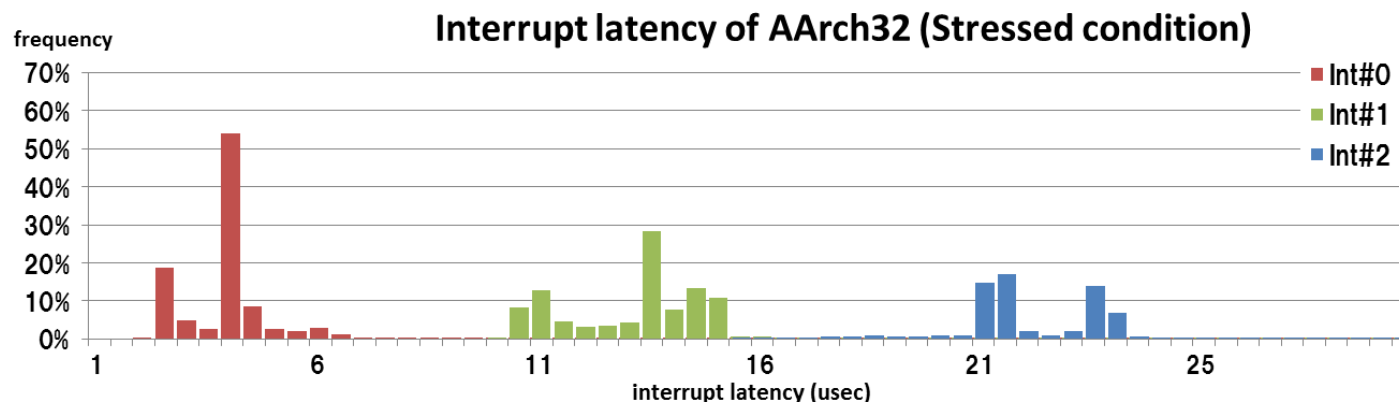
- Interrupt stack is implemented in AArch64 kernel 4.4
- We assumed that improvement for cache hit rate of the interrupt handler affect the improvement for interrupt latency, but the measurement results are different
- Memory transfer performance is different in addition to the difference of DMIPS, there is a possibility that it is affecting the performance difference between AArch32 and AArch64



# Response performance of interruption : Stressed condition

## ◆ Interrupt latency AArch32 vs AArch64 in stressed condition

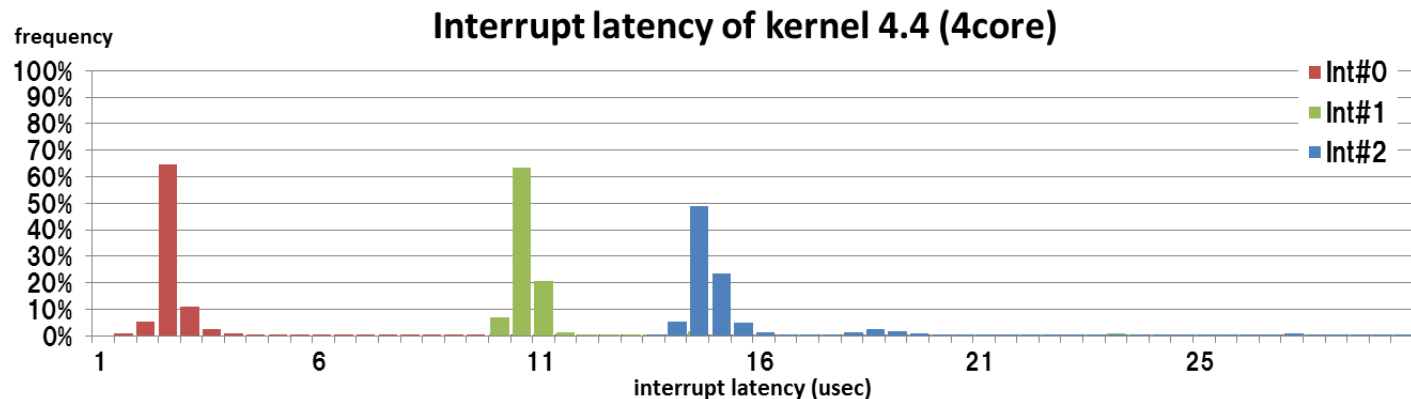
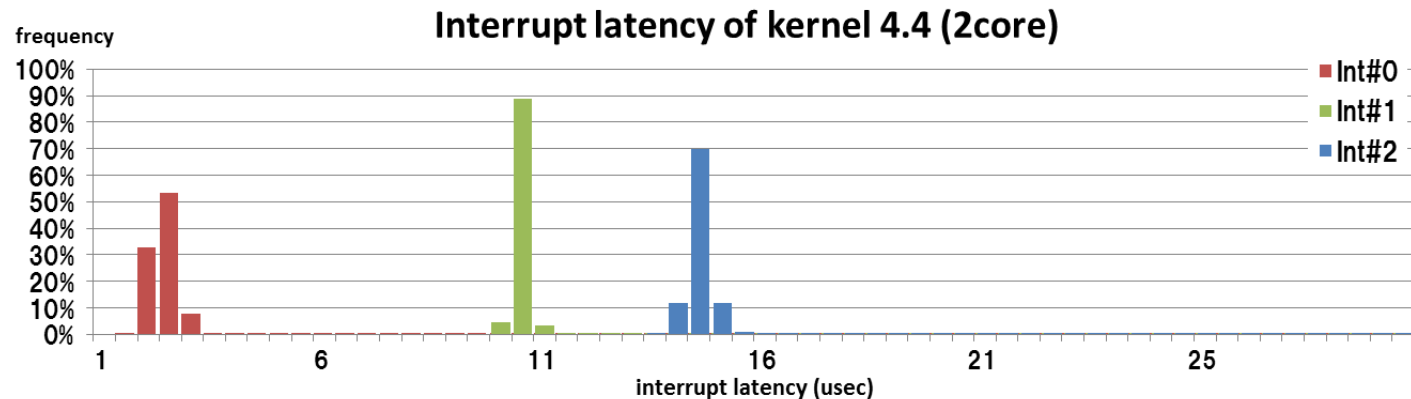
- Measure the IRQ vector latency with heavy processing load
- Process start-up was repeated with holding more than 80% CPU usage
- Deviation is larger on AArch32, but much not different on AArch64



# Response performance of interruption : 2core vs 4core

## ◆ Interrupt latency 2core vs 4core in stressed condition

- Measure the latency with changing the number of cores for the purpose of investigation for resistance to processing stress of AArch64
- Interrupt processing is linked to the 1core, but examined the possibility of change for CPU usage
- In consequence All CPUs got heavy stress correctly, and there is no marked difference for latency

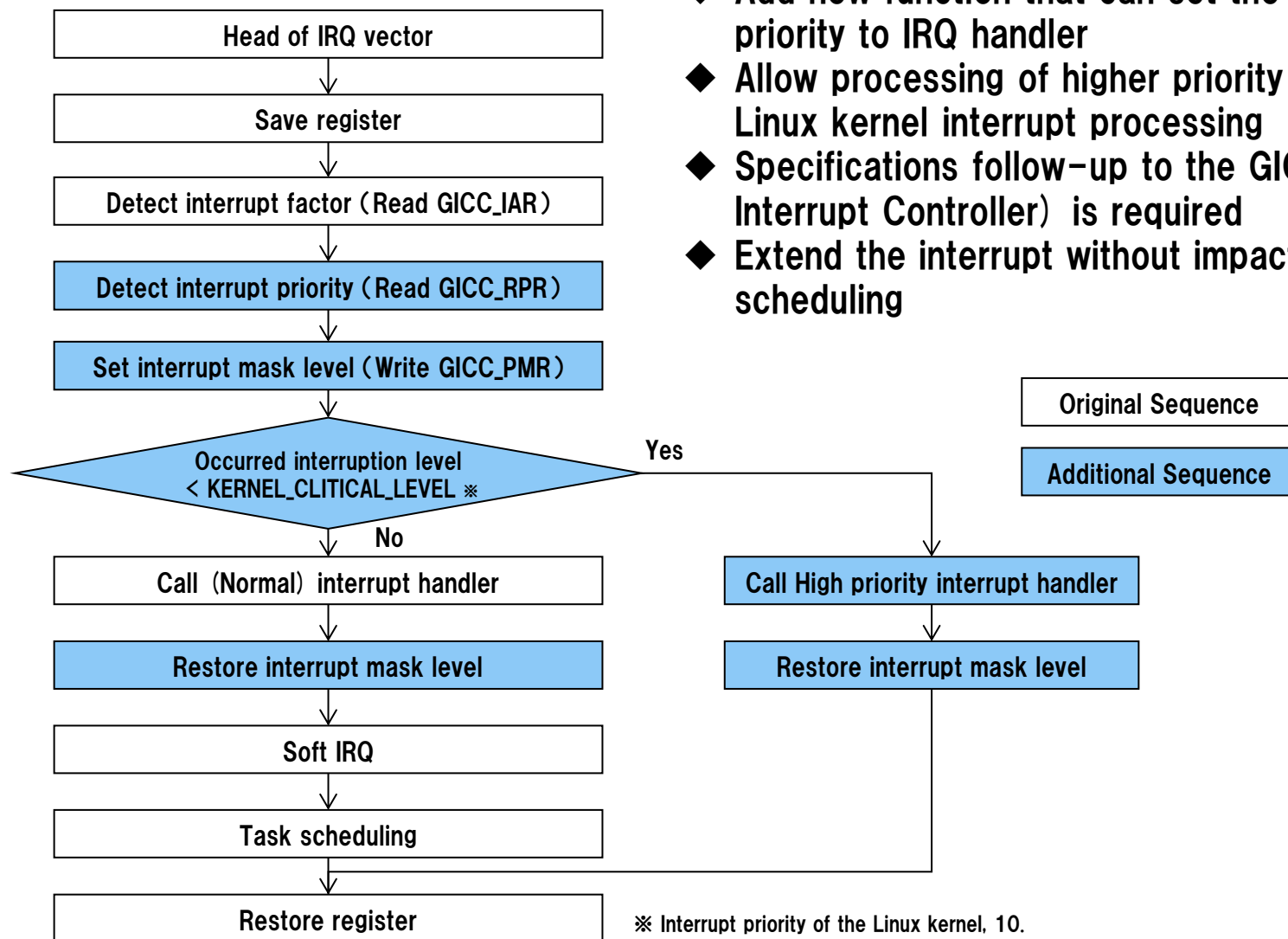


# Why latency improvement is occurred?

- ◆ why latency improvement for processing stress is occurred?
  - Investigate the state just before interrupt occurs
    - User applications are more often on AArch64
      - Measure the CPU usage with separation of kernel and application, but there is no marked difference
      - There may be a difference in the length of the critical section
  - “page\_fault” is often occurred on AArch32
    - There is a high possibility that the difference of Memory management has been affected
    - do\_fault\_around () was added on kernel 3.14, there is a high possibility that this function affects to improvement

- ◆ Basic interrupt performance has improved in AArch64
- ◆ However, connected equipment will increase more
  - Kind of approach of the interrupt response time improvement
    1. PREEMPT\_RT patch
      - Enabling kernel preemption
      - LTS kernels are supported by “Real-Time Linux”
    2. Interrupt priority
      - Adopted in the UNIX and RTOS
      - A thin layer between the Linux Kernel (SVC mode/EL1) and Hypervisor (HYP mode/EL2)
        - ✓ Small processing cost (Simple interrupt processing)
        - ✓ Small impact for the Linux kernel (It does not affect the scheduling process)
        - ✓ It can behave as light virtualization, functionality is obtained

# Overview of Interrupt priority



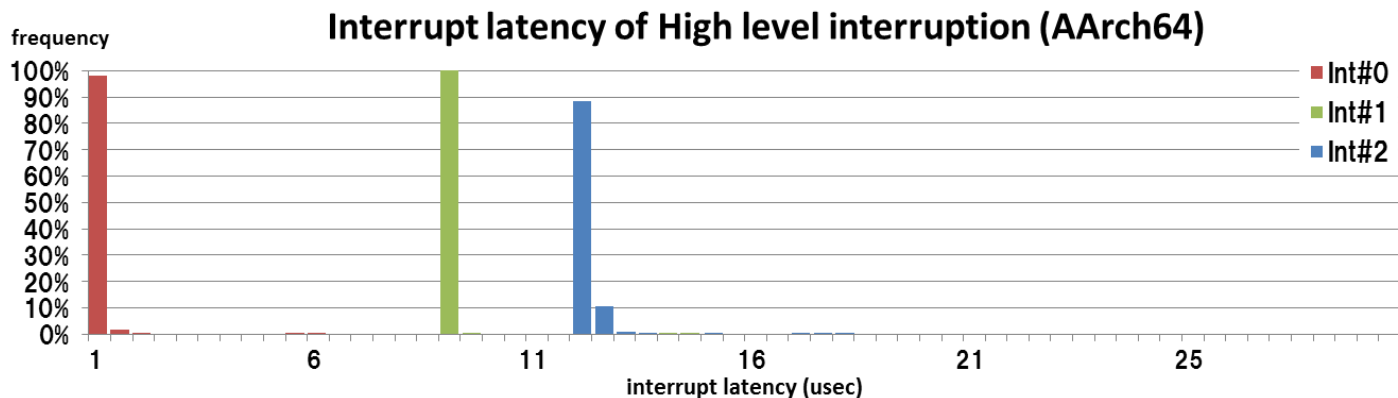
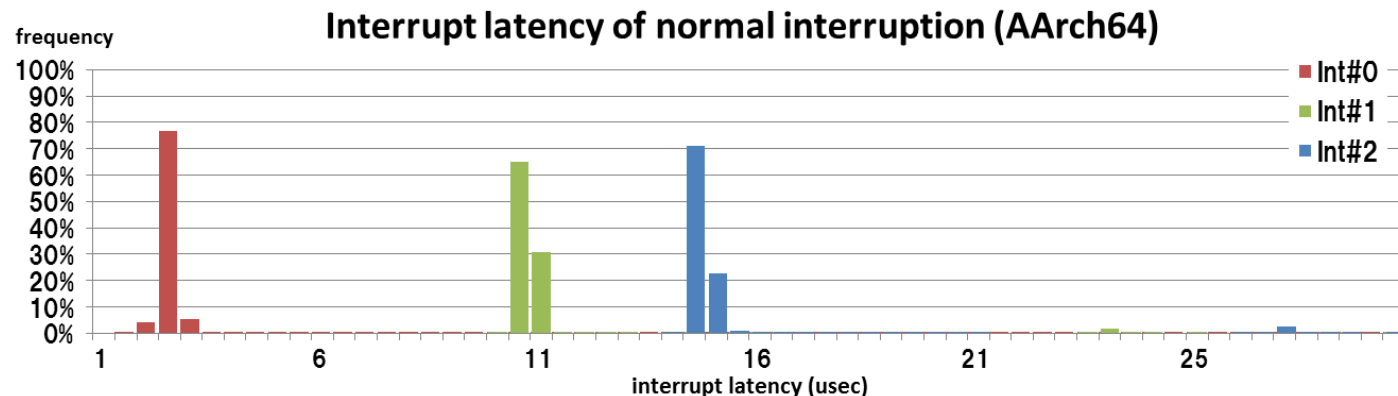
- ◆ Add new function that can set the interrupt priority to IRQ handler
- ◆ Allow processing of higher priority interrupts for Linux kernel interrupt processing
- ◆ Specifications follow-up to the GIC (Generic Interrupt Controller) is required
- ◆ Extend the interrupt without impact to the task scheduling



# Performance effect of Interrupt priority (64bit)

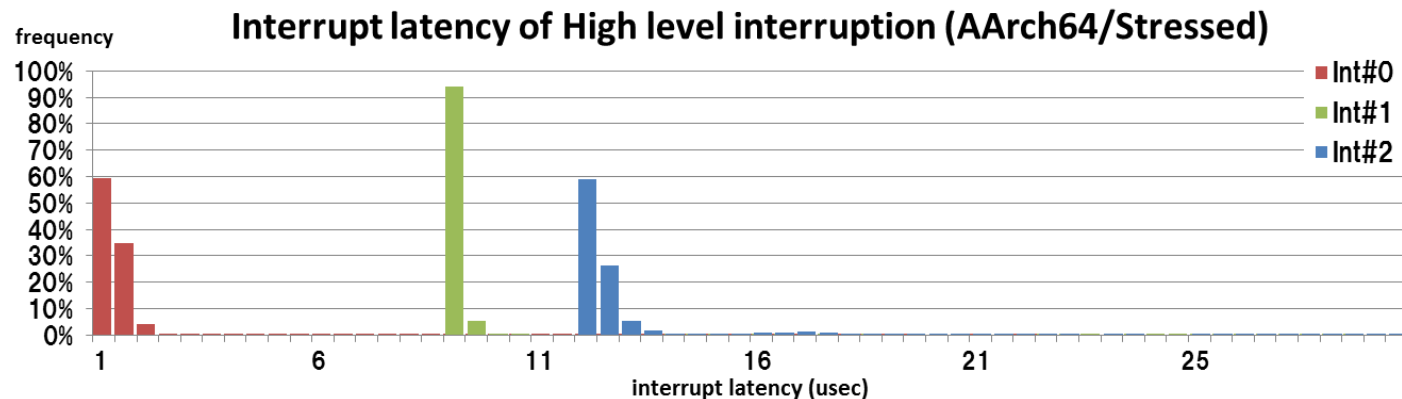
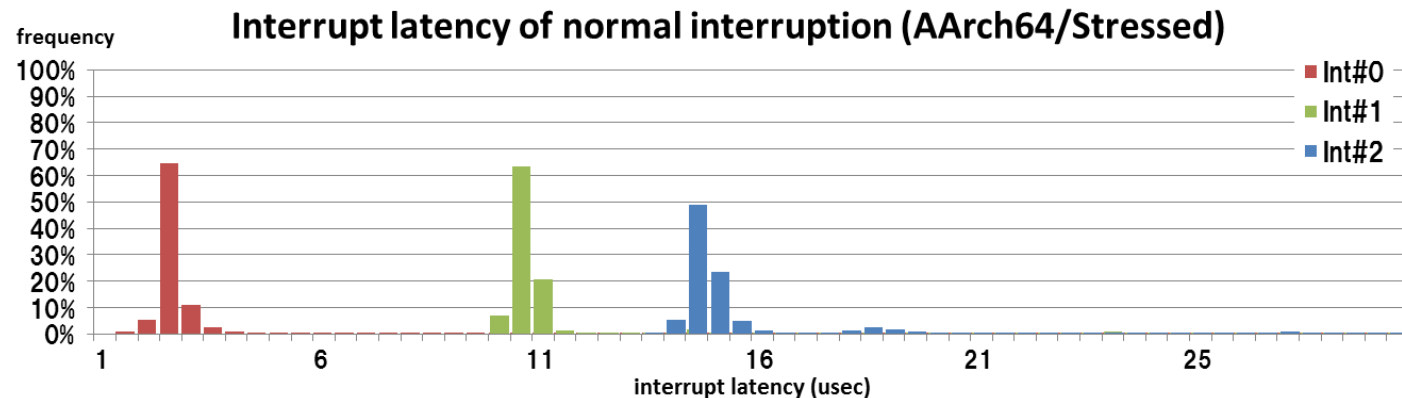
## ◆ Performance effect of Interrupt priority for AArch64

- Typical latency have improved about 20% on average
- It can be processed without effect of kernel critical section, so it is possible to implement the truly necessary processing with high priority
- However, the high priority interrupt became stronger than Linux kernel scheduling, it is necessary to be limited to the necessary processing



# Performance effect of Interrupt priority (64bit-stressed)

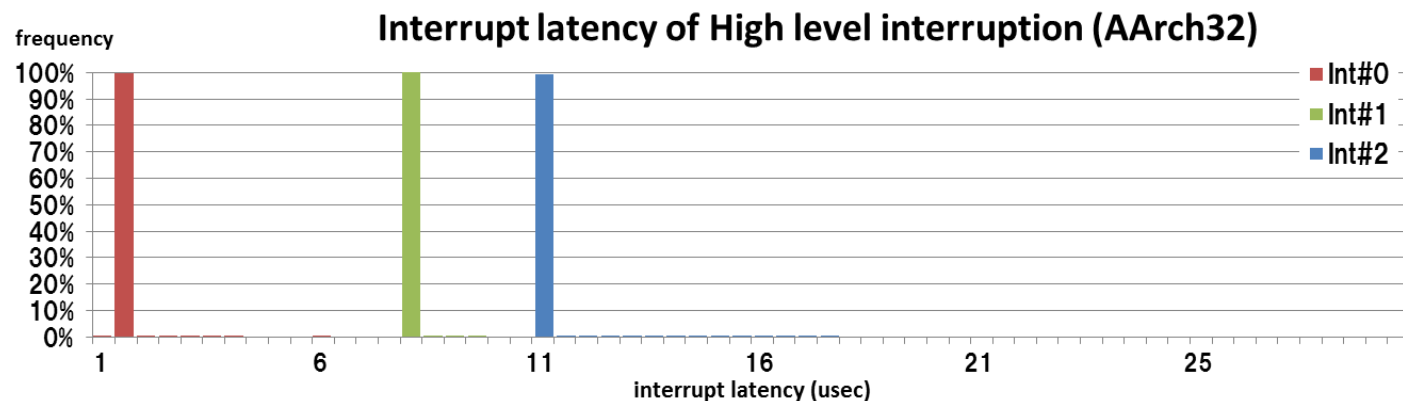
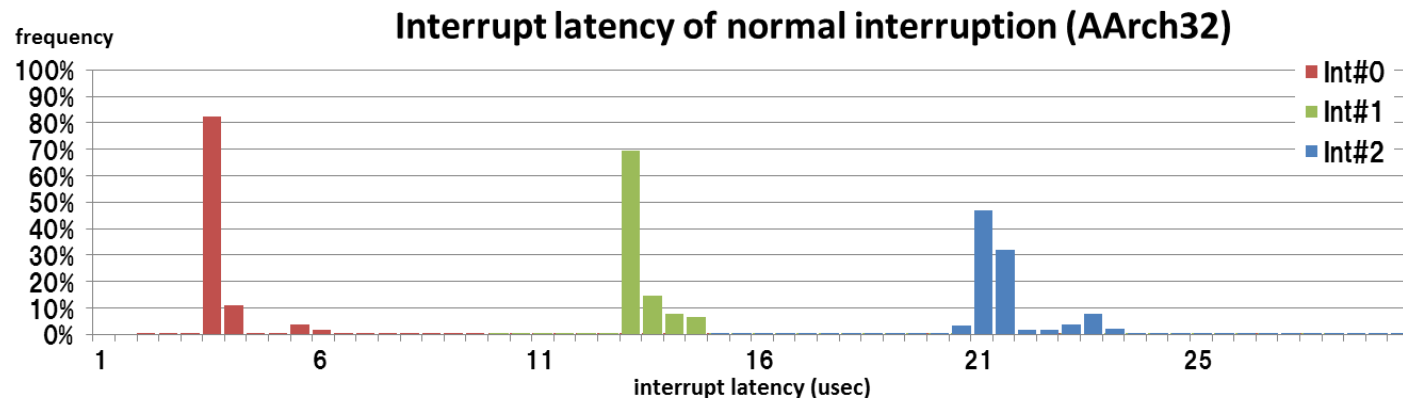
- ◆ Performance effect of Interrupt priority for AArch64 in stressed condition
  - It has been similar to the normal state improvement
  - Deviation became large in comparison with the normal state, but the average and typical latency has improved both



# Performance effect of Interrupt priority (32bit)

## ◆ Performance effect of Interrupt priority for AArch32

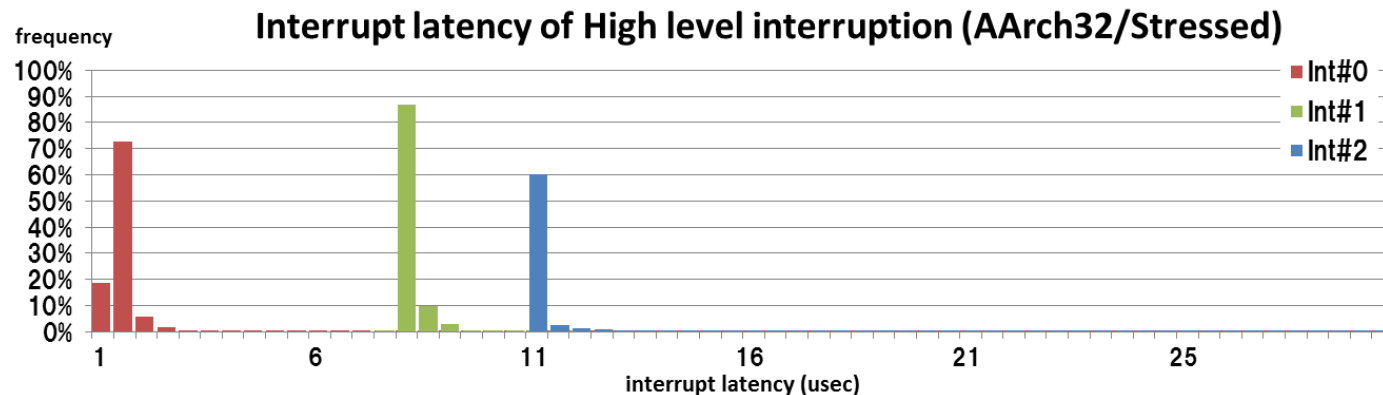
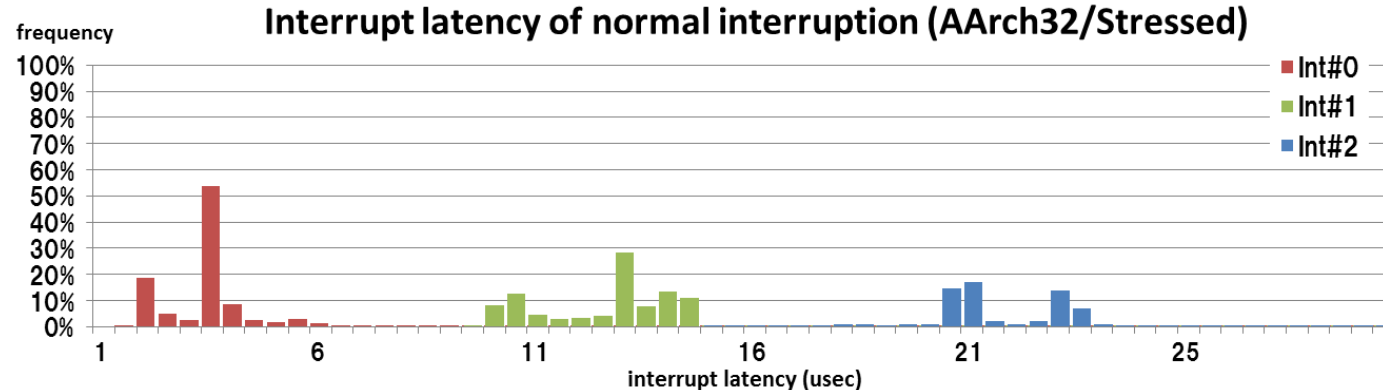
- Effect of level interruption was higher in AArch32, Typical latency have improved about 40% on average
- This indicates that there is still room for more improvement for AArch32 than AArch64



# Performance effect of Interrupt priority (32bit-stressed)

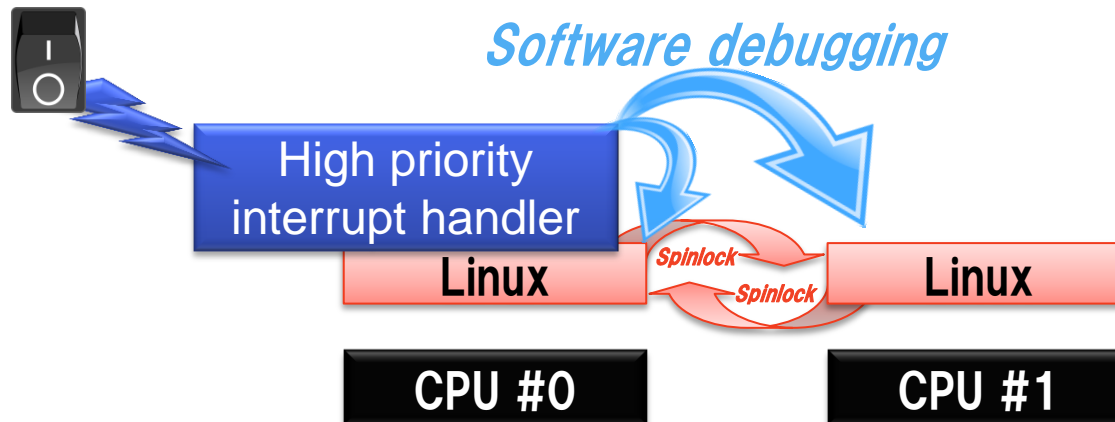
## ◆ Performance effect of Interrupt priority for AArch32

- Effect of level interruption was higher in AArch32 same as the normal condition
- Against the state in which the critical section is frequently, interrupt priority is effective to set the priority processing with low cost



# Functional effect of level interruption

- ◆ Possible to debug of the locked kernel without JTAG debugger
  - Can interrupt in the Linux kernel that was stopped in Spinlock
  - Can debug the Linux kernel if implement the console to the handler
  - Kernel functions are not available, Generic debug also possible by combination with kgdb (Modification of kgdb is needed)
  - There is no need for expensive JTAG debugger, All developers can use



# Summary

- ◆ AArch64 have advantage in terms of the processing performance and memory capacity in comparison to AArch32
- ◆ OSS that can be build for AArch64 is increasing day by day
- ◆ There is no universal solution to the confliction of Tool chain, steady setting for “configure” is necessary
- ◆ 32-bit application may not work on AArch64
  - ◆ Correct code to work correctly, but bad manners code does not work
  - ◆ Static analysis is effective for AArch64
- ◆ Interrupt response is also improved on AArch64
- ◆ Newer Linux kernel has better resistance to processing stress
- ◆ For further response improvement, Interrupt priority is effective
  - ◆ Resistance to processing stress was enhanced on AArch32/kernel 3.10
  - ◆ Because it can operate outside of the kernel management, it is possible to debug the kernel

◆ **ARM 64-bit Architecture**

[https://events.linuxfoundation.org/images/stories/pdf/lcna\\_co2012\\_marinas.pdf](https://events.linuxfoundation.org/images/stories/pdf/lcna_co2012_marinas.pdf)

◆ **Public service announcement to help 64-Bit ARM port**

<https://wiki.linaro.org/RikuVoipio/TemplateRequest>

◆ **Splint**

<http://www.splint.org/>

◆ **Real-Time Linux Wiki**

[https://rt.wiki.kernel.org/index.php/Main\\_Page](https://rt.wiki.kernel.org/index.php/Main_Page)

# Thank you!