FUJITSU

shaping tomorrow with you

Kernel Interrupt: A Major Overhaul

- APIC Initialization

&

- Vector Allocation

Dou Liyang douly.fnst@cn.fujitsu.com

June 20 2018

Copyright 2018 FUJITSU LIMITED

Outline

FUJITSU

What's next?

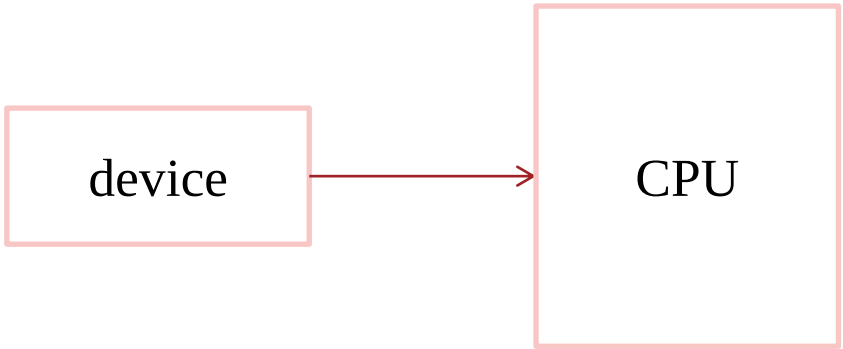
- ***Basics of an interrupt***
- Overhaul of interrupt
  - APIC Initialization
  - Vector Allocation
- Future work

Copyright 2018 FUJITSU LIMITED

What is An Interrupt?

FUJITSU

- A hardware signal
- Emitted from a peripheral to a CPU
- Indicating that a device-specific condition has been satisfied



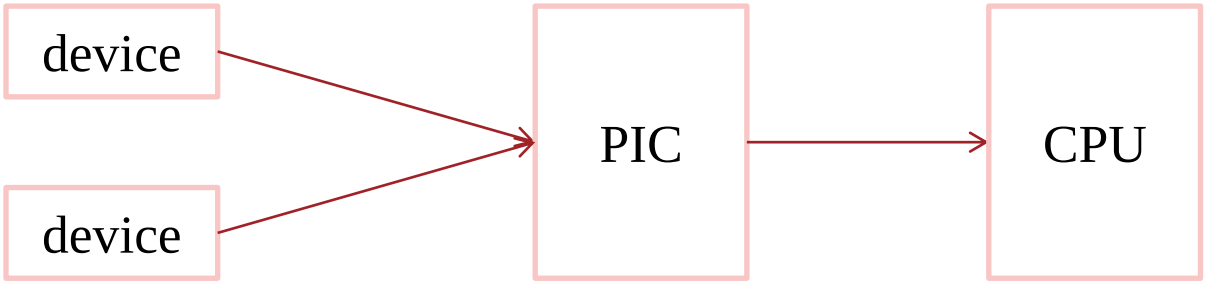
From Marc Zyngier <marc.zyngier@arm.com>

Copyright 2018 FUJITSU LIMITED

Multiplexing Interrupts

FUJITSU

- Having a single interrupt for the CPU is usually not enough
- Most systems have tens, hundreds of them
- ***An interrupt controller*** allows them to be multiplexed
- Very often architecture or platform specific
- In old x86 machine, there was a PIC called 8259A
  - a chip responsible for sequentially processing multiple interrupt requests from multiple devices
  - Called ***PIC Mode***



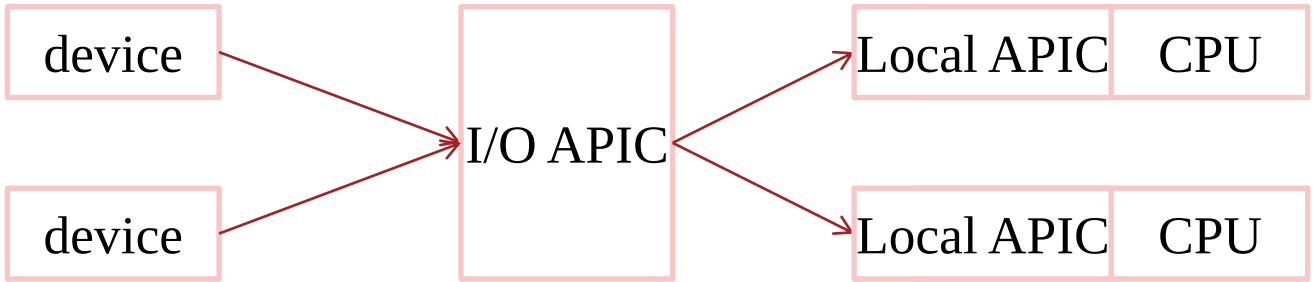
Copyright 2018 FUJITSU LIMITED

Multiplexing Interrupts in SMP System

FUJITSU

- Only a CPU is usually not enough
- Most systems have tens, hundreds of CPUs
- An new interrupt controller should be used
- In x86 machine, there is an ***APIC***
  - Local APIC is located on each CPU core, handles the CPU-specific interrupt configuration
  - I/O APIC distribute external interrupts from multiple devices to multiple CPU cores

Symmetric I/O Mode



Copyright 2018 FUJITSU LIMITED

More than wired interrupts: MSIs

FUJITSU

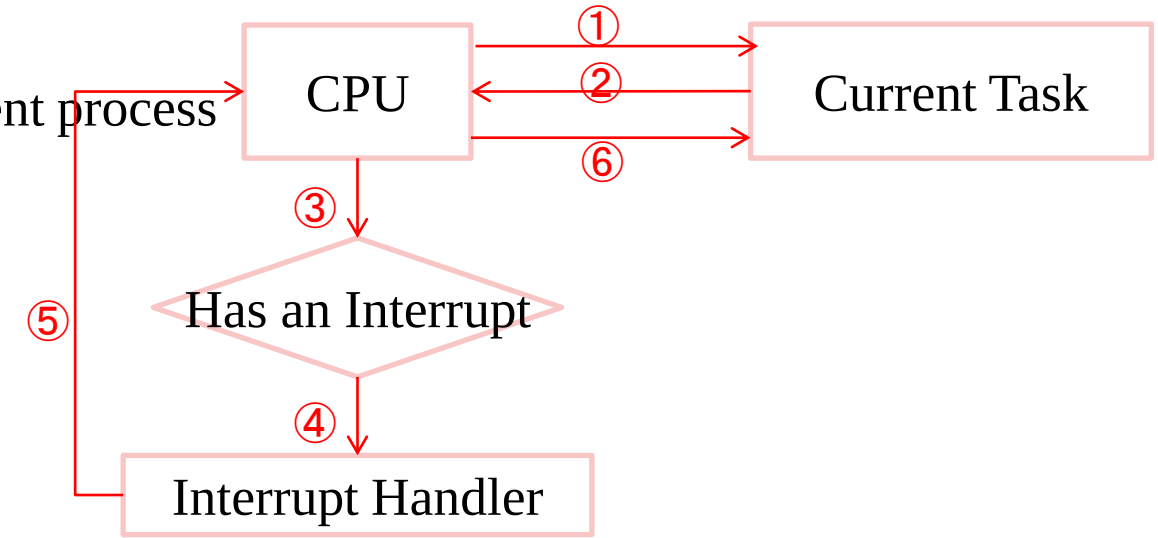
- ***Message Signaled Interrupts*** are as an alternative to line-based interrupts
  - Trigger an interrupt by writing a value to a particular memory
  - Allow the use of the same buses as the data

Copyright 2018 FUJITSU LIMITED

Handle an Interrupt

FUJITSU

- Preempt current task ① ②
  - ***Pause execution*** of the current process.
- Execute interrupt handler ③ ~ ⑤
  - Search for ***the handler of the interrupt*** and transfer control
- Resume the task ⑥
  - ***Return to execute*** the current process

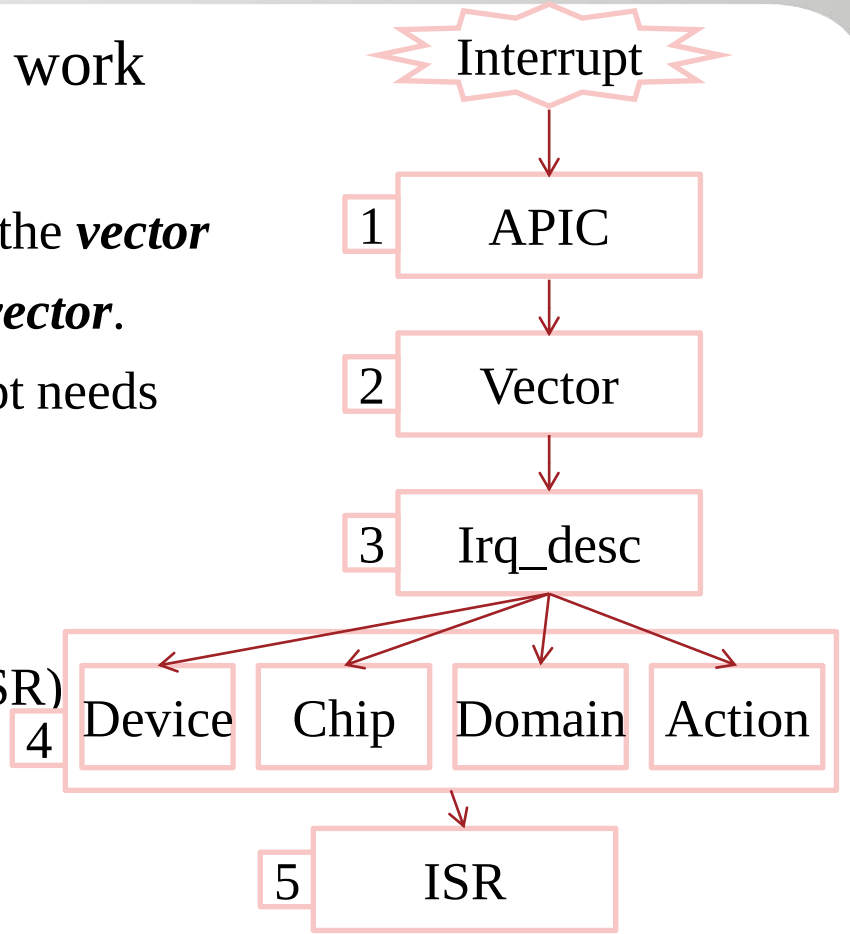


Copyright 2018 FUJITSU LIMITED

How Does “Handle an Interrupt” Work?

FUJITSU

- ***APIC*** and ***Vector*** mechanism make it work
  1. Delivery the IRQ through the ***APIC***
  2. CPU search the handler in IDT through the ***vector***
  3. Get the ***irq_desc*** structure through the ***vector***.
  4. Use the irq_desc to get what the interrupt needs
     - ***device*** info
     - ***interrupt controller*** info
     - ***IRQ action*** list info
  5. Execute the interrupt service routine (ISR)



Copyright 2018 FUJITSU LIMITED

Why "*APIC* and *Vector* " Can Work?

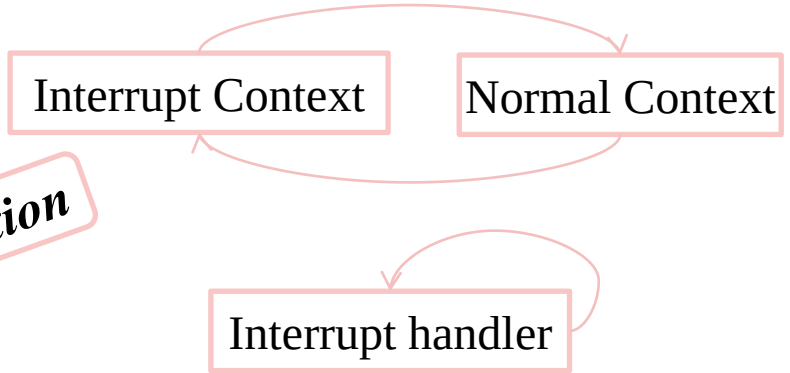
FUJITSU

- ***Do many initialization and setup works*** when Linux boots up
  - For the interrupt delivery
    - Initialize 8259A
    - Switch ***interrupt delivery mode***
    - Initialize APIC
      - Set Local APIC & I/O APIC

APIC Initialization

  - For IDT table
    - Initialize the mapping of Vector and Handler
  - For each Interrupt
    - Allocate an IRQ
    - Allocate an irq_desc
    - Assign a vector

Vector Allocation



Copyright 2018 FUJITSU LIMITED

Outline

FUJITSU

What's next?

- Basics of an interrupt
- ***Overhaul of interrupt***
  - ***APIC Initialization***
  - Vector Allocation
- Future work

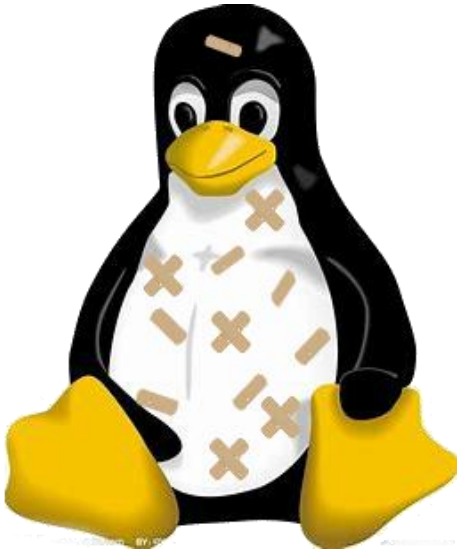
Copyright 2018 FUJITSU LIMITED

Existing Problems

FUJITSU

- Interrupt in x86 is a ***conglomerate*** of ancient bits and pieces
  - Subject to '***modernization***' and ***features*** over the years
    - *Kdump*
    - CPU Hotplug/System hibernation
    - Multi-queue devices

- It looks like a penguin full of band-aids
  - Can work, but can’t see how it works easily.

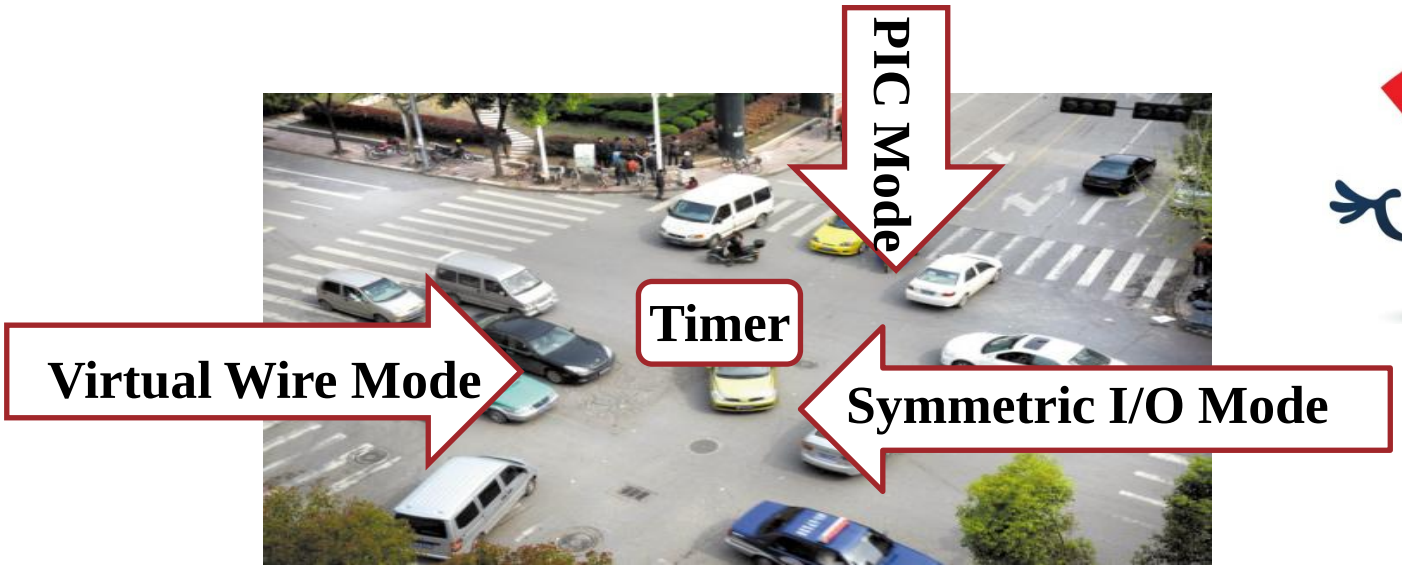


Copyright 2018 FUJITSU LIMITED

Problems of APIC Initialization

FUJITSU

- Horrible interrupt mode setup
  - Setup the mode **at random places**
  - Run the kernel with **the potentially wrong mode**
- Tangle the timer setup with interrupt initialization

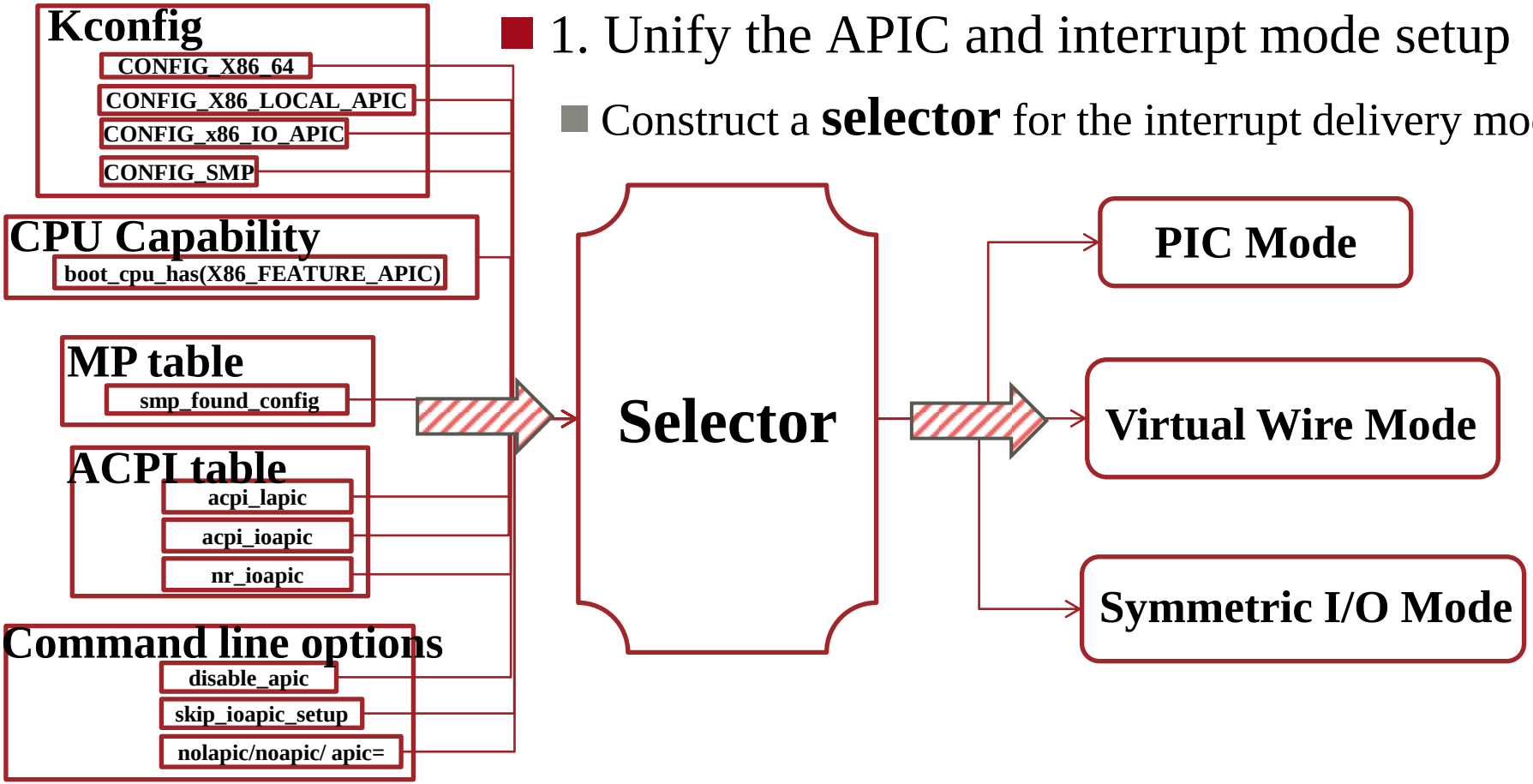


Copyright 2018 FUJITSU LIMITED

Overhaul of APIC Initialization

FUJITSU

- 1. Unify the APIC and interrupt mode setup
  - Construct a **selector** for the interrupt delivery mode

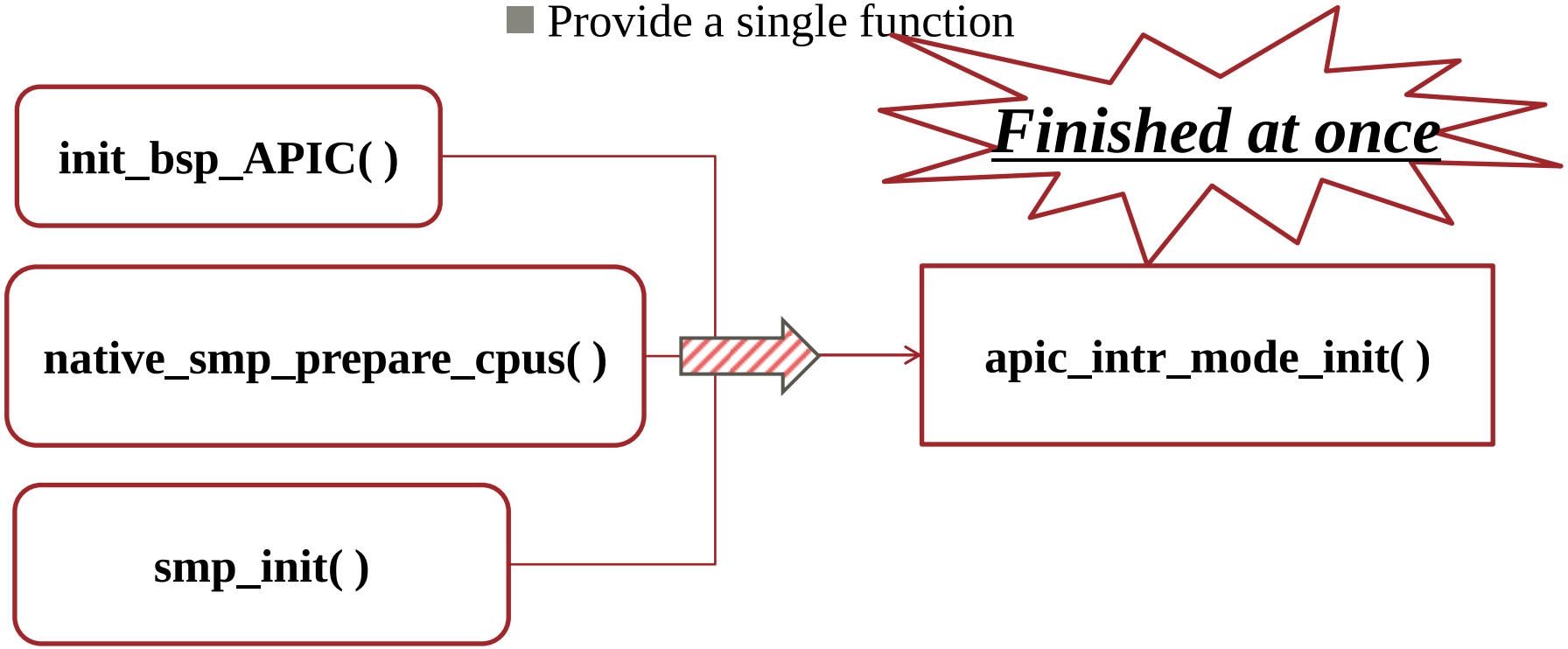


Copyright 2018 FUJITSU LIMITED

Overhaul of APIC Initialization

FUJITSU

- 1. Unify the APIC and interrupt mode setup
  - Provide a single function

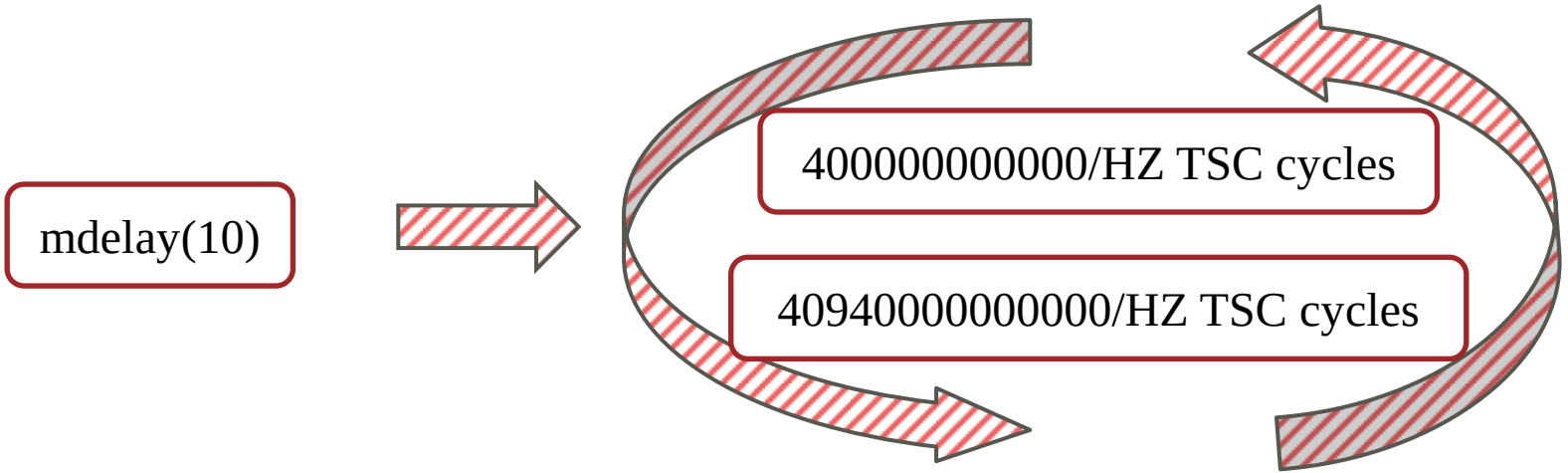


Copyright 2018 FUJITSU LIMITED

Overhaul of APIC Initialization

FUJITSU

- 2. Disentangle the timer setup from the APIC initialization
  - Refactor **the delay logic** during APIC initialization process.
    - Either use TSC or a simple delay loop to make a rough delay estimate



  - Split local APIC timer setup from the APIC setup

Copyright 2018 FUJITSU LIMITED

Overhaul of APIC Initialization

FUJITSU

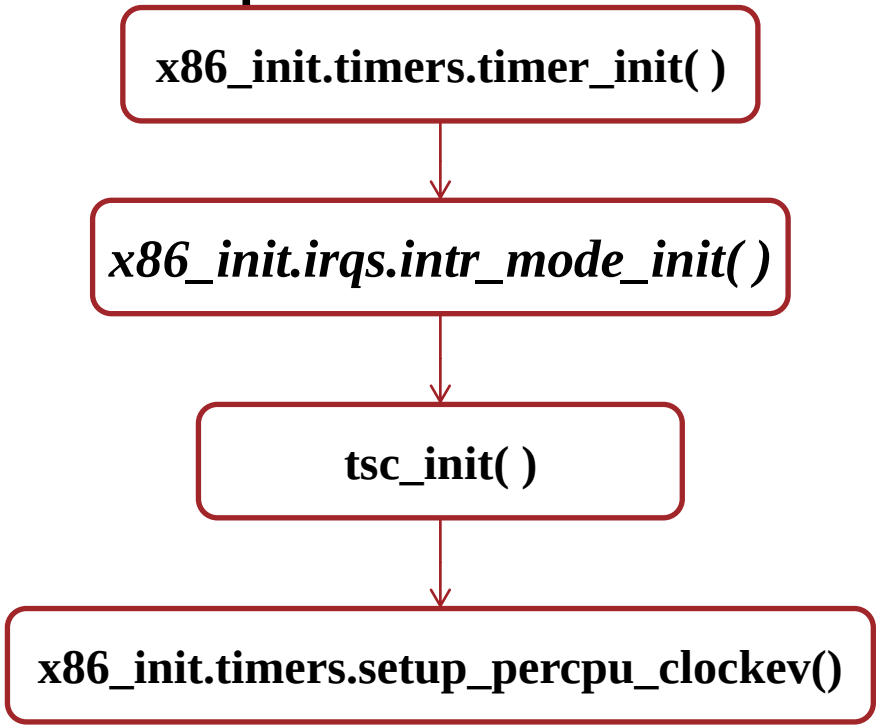
- 3. **Reorganize** the interrupt initialization
  - Set up the final interrupt delivery mode **as soon as possible**.

1) Set up the legacy timer(PIT/HPET)

2) Set up APIC/IOAPIC

3) TSC calibration

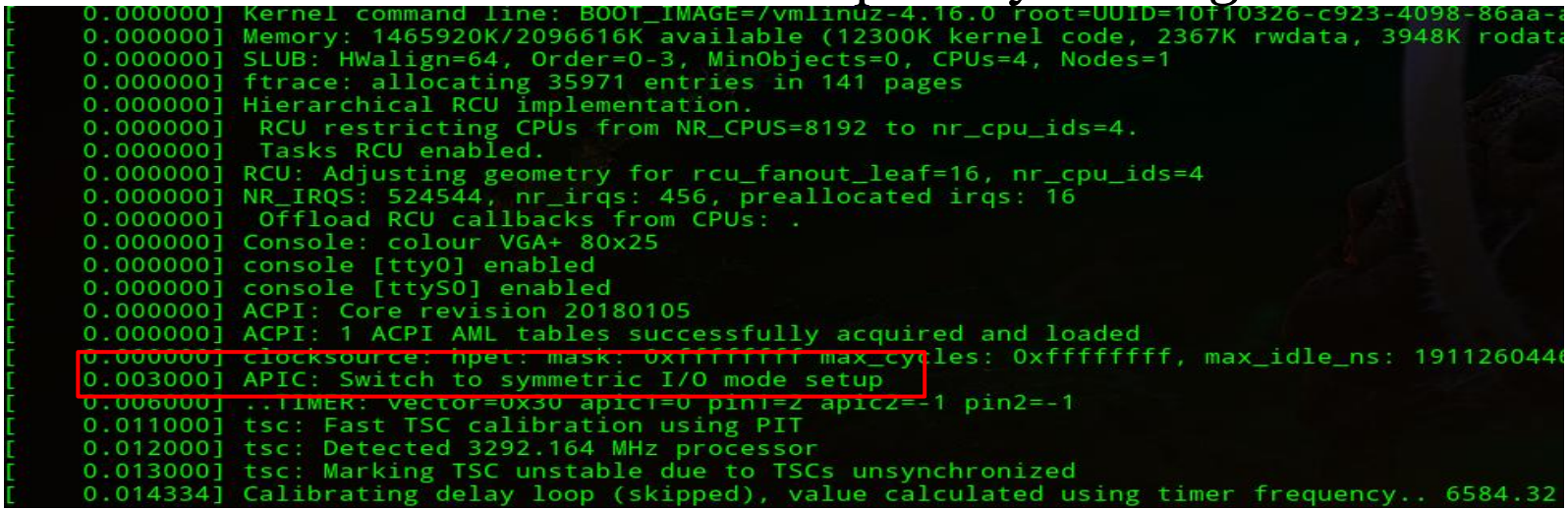
4) Local APIC timer setup



Copyright 2018 FUJITSU LIMITED

Overhaul of APIC Initialization

- 4. Some others
  - Refactor some common APIC function
  - Compatible with ACPI initialization
  - Bypass the hypervisor, Such as KVM and Xen
- 5. Can check which mode the interrupt is by ‘dmesg’:



Copyright 2018 FUJITSU LIMITED

Outline

FUJITSU

What's next?

- Basics of an interrupt
- ***Overhaul of interrupt***
  - APIC Initialization
  - ***Vector Allocation***
- Future work

Copyright 2018 FUJITSU LIMITED

Problems of ***Vector Allocation***

FUJITSU

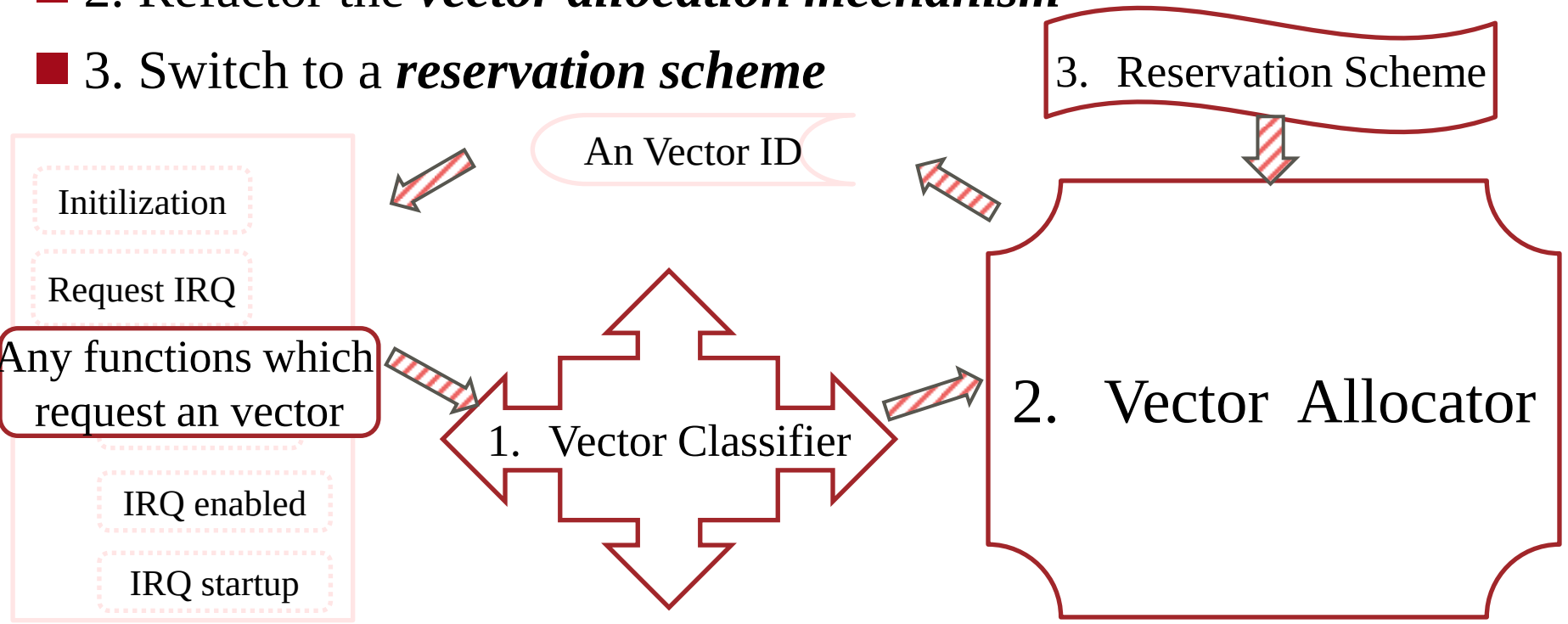
- Horrible worst vector management mechanism
  - ***Abuse*** the interrupt allocation for different type interrupts
  - Serve all different use cases ***in one go***
  - Based on ***nested loops*** to search
  - Cause vector space ***exhaustion***
  - Allocate vectors at the ***wrong*** time and on the ***wrong*** place
- Some dubious properties, causes ***high complexity***
  - Multi CPU affinities for an IRQ
  - Priority level spreading
- Lack of instrumentation
  - All of this is a black box which allows no insight into the actual vector usage

Copyright 2018 FUJITSU LIMITED

Overhaul of ***Vector Allocation***

FUJITSU

- 1. Classify the ***types*** of vectors
- 2. Refactor the ***vector allocation mechanism***
- 3. Switch to a ***reservation scheme***

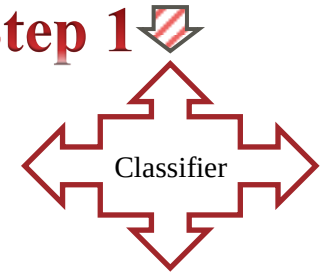


Copyright 2018 FUJITSU LIMITED

Overhaul of ***Vector Allocation***

FUJITSU

- 1. Classify the ***types*** of vectors
- Each CPU has 256 vectors, But some are ***fixed***
  - ***1. System Vector***
    - ***Vectors 0 ... 31***
    - ***Vector 128***
    - ***Vectors INVALIDATE_TLB_VECTOR_START ... 255***
  - ***2. Legacy Vector***
    - ***Vectors 0x30 ... 0x3f***
  - **Others** are allocated dynamically for ***normal*** and ***managed*** interrupts.



Copyright 2018 FUJITSU LIMITED

Overhaul of ***Vector Allocation***

FUJITSU

- 1. Classify the ***types*** of vectors
  - For external interrupts
  - Depend on ***Interrupt Affinity***(the set of CPUs that can handle this interrupt)
    - ***3. Normal Vector***
    - ***4. Managed Vector***

	Normal Interrupt	***Managed Interrupt***
At setup time	- Affinity may be NULL - A subset of the online CPUs	- Affinity must have been setup - the possible CPUs may be included
User space	- Affinity can be modified	- Affinity is fixed
When migration	- IRQ can be moved to any online CPUs - Affinity can be even reset	- IRQ can move only in the affinity. - But, can be shutdown and restarted. - Affinity can't be reset

Copyright 2018 FUJITSU LIMITED

Overhaul of *Vector Allocation*

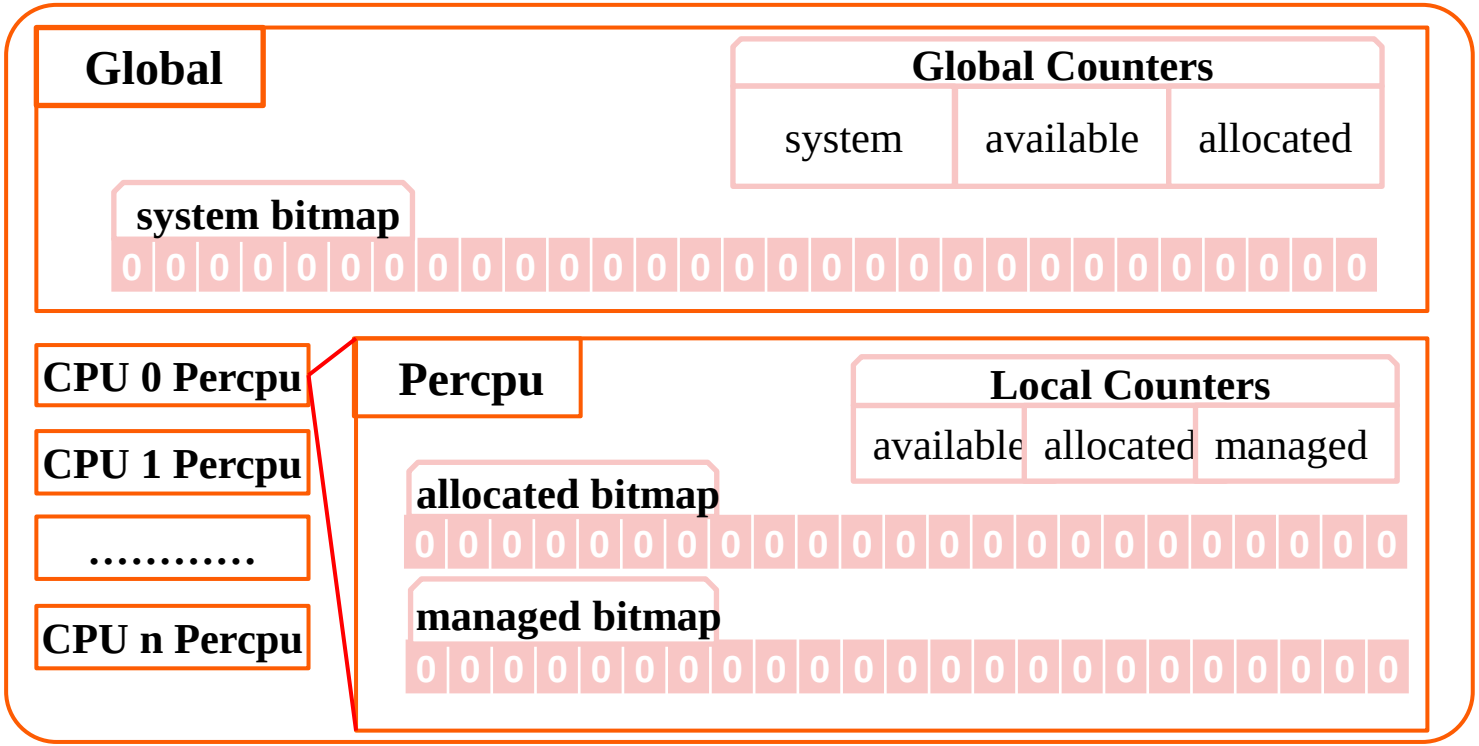
FUJITSU

2. Refactor the *vector allocation mechanism*

Step 2

Allocator

- Create a new bitmap matrix allocator——*IRQ Matrix*

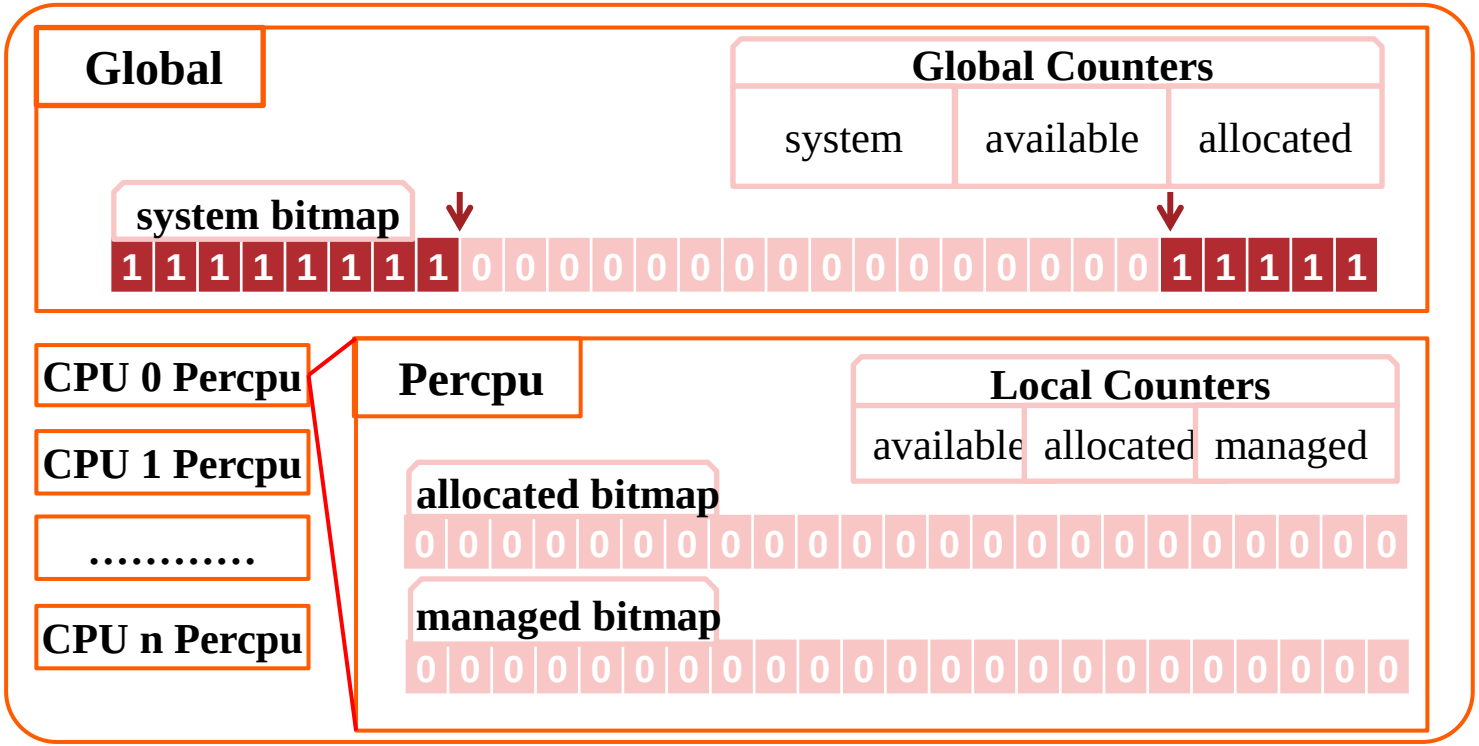


Copyright 2018 FUJITSU LIMITED

Overhaul of *Vector Allocation*

FUJITSU

- 2. Refactor the vector allocation mechanism
  - Use the matrix for ***System*** vector

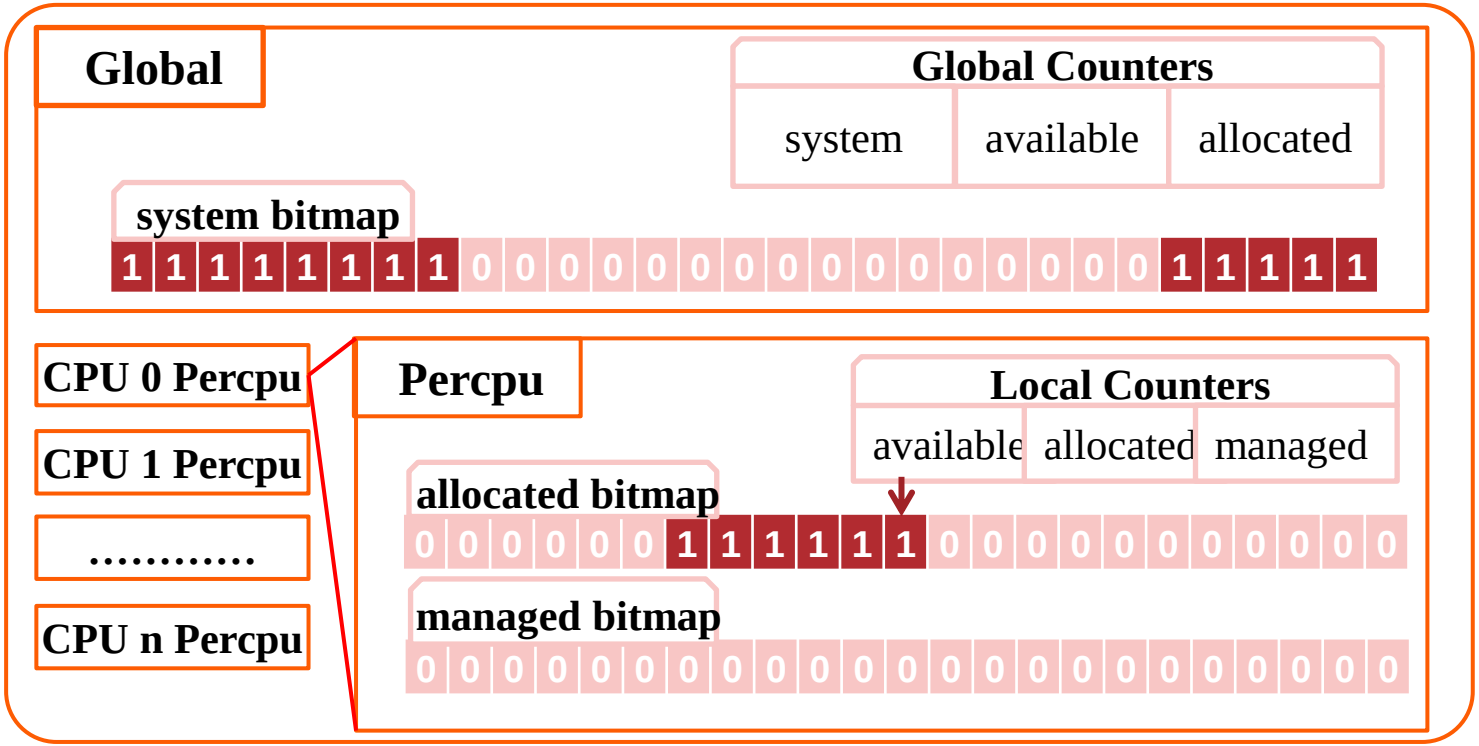


Copyright 2018 FUJITSU LIMITED

Overhaul of *Vector Allocation*

FUJITSU

- 2. Refactor the vector allocation mechanism
  - Use the matrix for ***Legacy*** vector

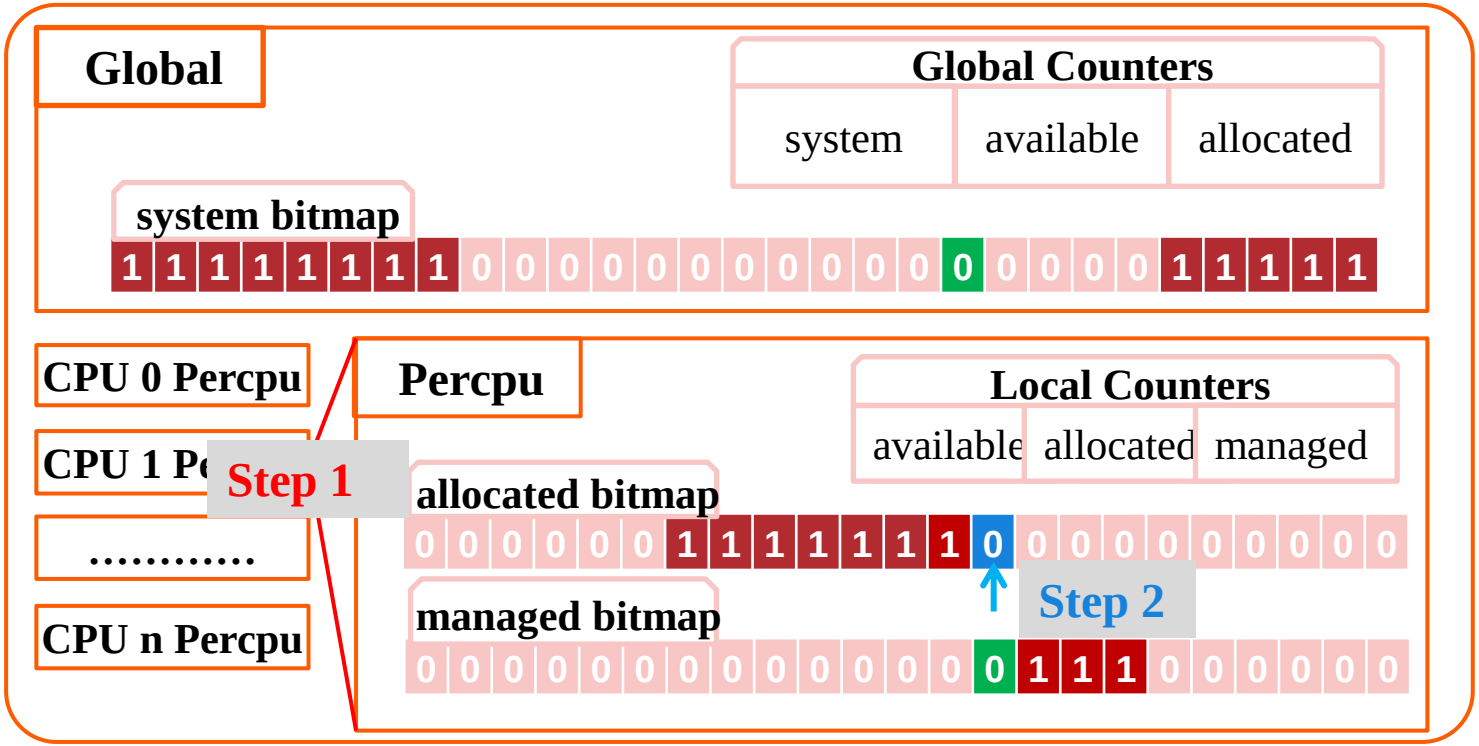


Copyright 2018 FUJITSU LIMITED

Overhaul of *Vector Allocation*

FUJITSU

- 2. Refactor the vector allocation mechanism
  - Use the matrix for ***Normal*** vector

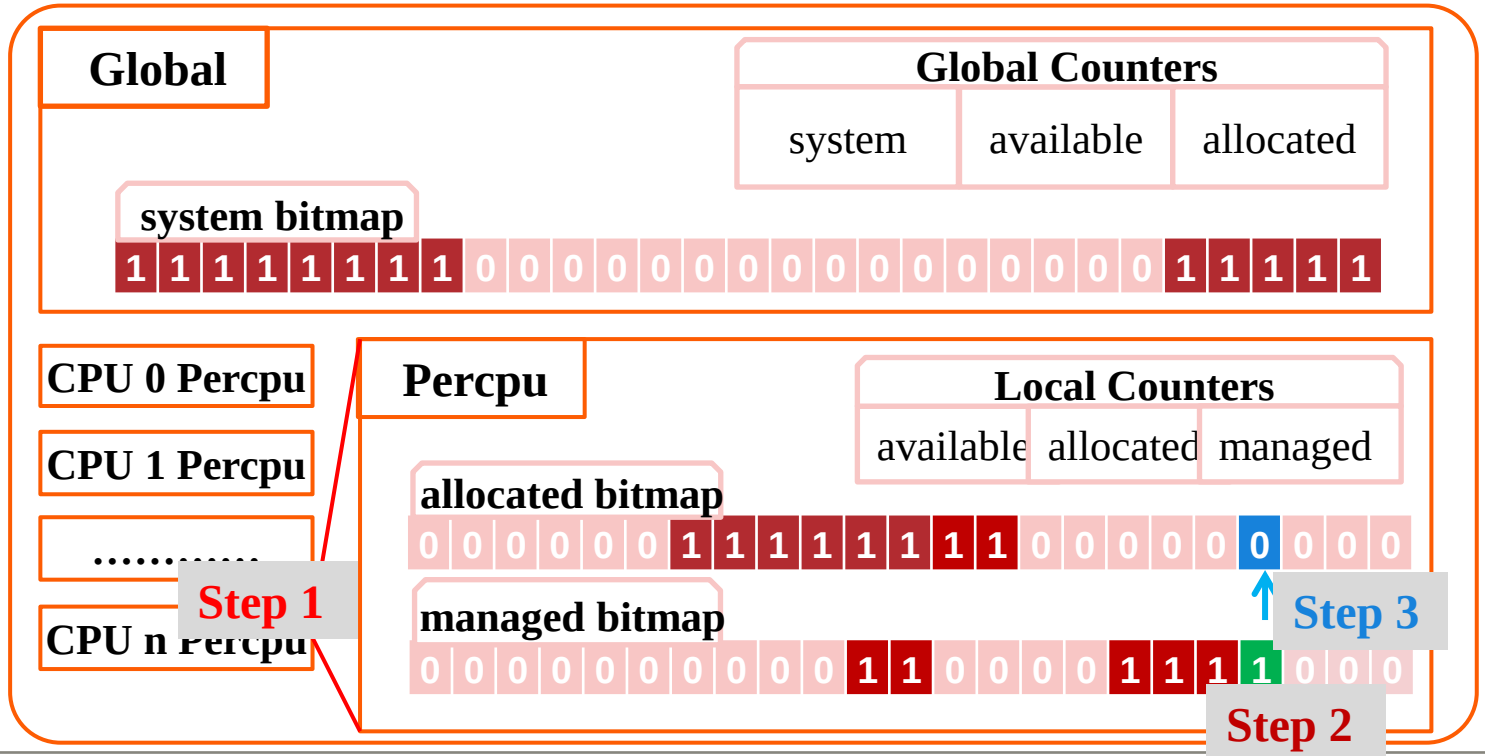


Copyright 2018 FUJITSU LIMITED

Overhaul of *Vector Allocation*

FUJITSU

- 2. Refactor the vector allocation mechanism
  - Use the matrix for ***Managed*** vector



Copyright 2018 FUJITSU LIMITED

Overhaul of ***Vector Allocation***

FUJITSU

3. Switch to ***reservation scheme***

- Reserve a ***new system vector*** , just in case

Step 3

Reservation

3.1 When the interrupt is allocated and initialized:

Previously

wasteful

Assign a ***real vector*** for each interrupts

Now

1. Update the ***reservation request counter***

2. Assign ***the reserved vector*** for each interrupts

Copyright 2018 FUJITSU LIMITED

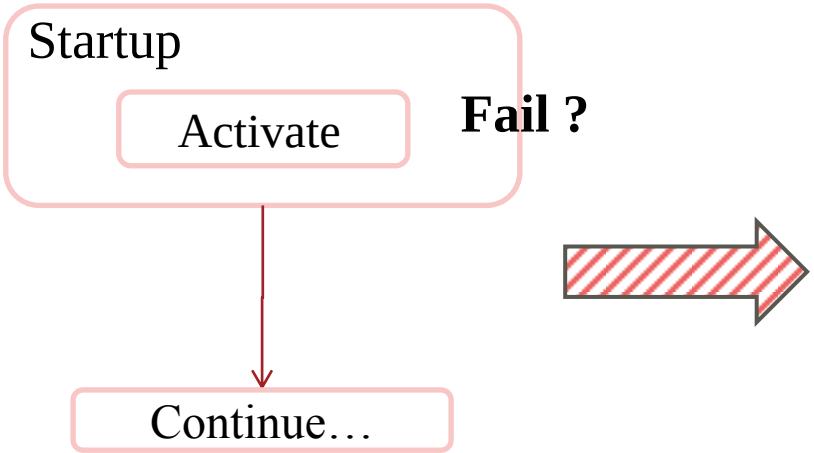
Overhaul of ***Vector Allocation***

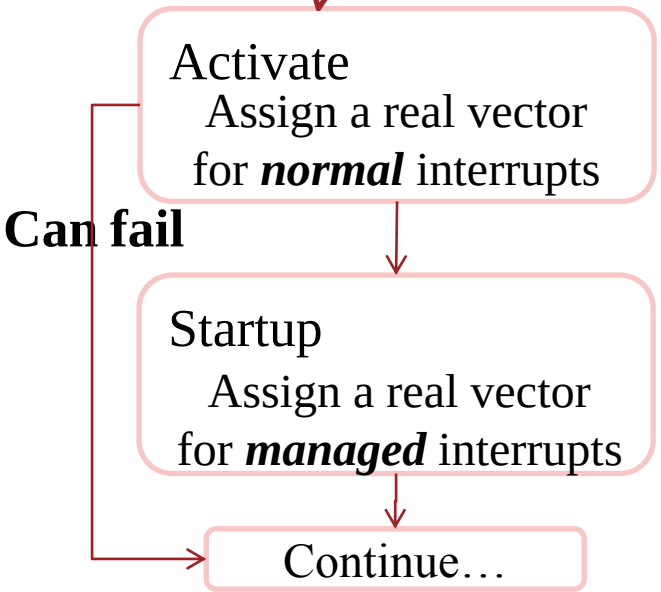
FUJITSU

Vector Space Saving

- 3. Switch to ***reservation scheme***
  - ***Separate*** activation and startup

3.2 When the interrupt is ***requested***:





Copyright 2018 FUJITSU LIMITED

Overhaul of *Vector Allocation*

FUJITSU

- Some Others:
  - Change from Multi CPU targets to ***single interrupt targets***.
  - Remove priority level spreading
  - Simplify hotplug vector accounting
  - Equip with trace points and detailed debugfs information

```
[root@guest_64bit ~]# cat /sys/kernel/debug/irq/domains/*
name:   VECTOR
 size:   0
 mapped: 25
 flags:  0x00000041
Online bitmaps:        2
Global available:    377
Global reserved:       4
Total allocated:      21
System: 43: 0-19,32,50,128,236-255
 | CPU | avl | man | act | vectors
     0   190     4    11  34-40,42-44,48
     1   187     4    10  33-37,40-42,45-46
name:   IO-APIC-0
 size:   24
 mapped: 15
 flags:  0x00000041
 parent: VECTOR
    name:   VECTOR
     size:   0
     mapped: 25
     flags:  0x00000041
Online bitmaps:        2
Global available:    377
Global reserved:       4
Total allocated:      21
System: 43: 0-19,32,50,128,236-255
     | CPU | avl | man | act | vectors
         0   190     4    11  34-40,42-44,48
         1   187     4    10  33-37,40-42,45-46
```

- Can see the Vector Allocation by:
  - cat /sys/kernel/debug/irq/irqs/$N
  - cat /sys/kernel/debug/irq/domains/$N

Copyright 2018 FUJITSU LIMITED

Outline

FUJITSU

What's next?

- Basics of an interrupt
- Overhaul of interrupt
  - APIC Initialization
  - Vector Allocation
- ***Future work***

Copyright 2018 FUJITSU LIMITED

Future work

FUJITSU

- Kernel's notion of possible CPU count should be realistic
  - Once the kernel initialized:

    Make the possible CPU count realistic

- The vector allocation is a ***generic*** mechanism
  - Can be used to other architectures

Copyright 2018 FUJITSU LIMITED

Thank you !

Copyright 2018 FUJITSU LIMITED