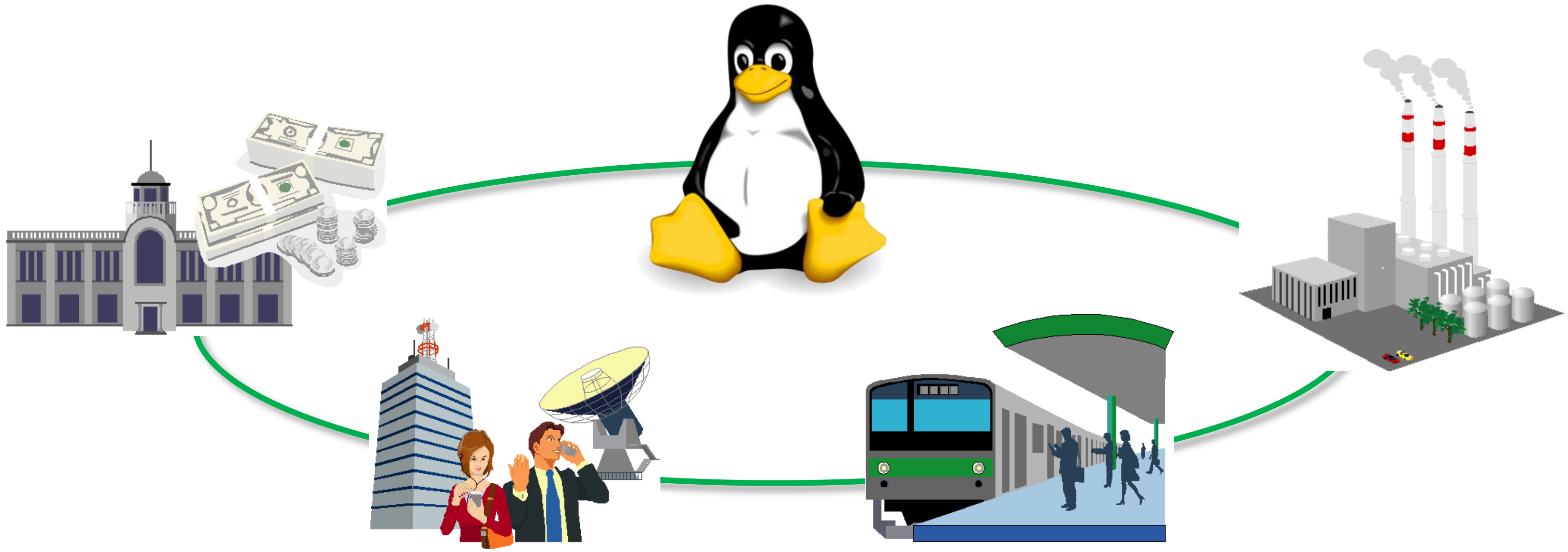
HITACHI
Inspire the Next

RAS Enhancement Activities for Mission-Critical Linux Systems

Hitachi Ltd.
Yoshihiro YUNOMAE

Yokohama Research Lab.
Linux Technology Center

0-1 Mission-Critical Systems



- We apply Linux to mission-critical systems.
  - Banking systems/carrier backend systems/train management systems and more
  - People(consumers/providers) expect stable operation for long-term use.
    - Don't frequently change the system configuration
    - Changing the system introduces the risk for illegal operation.
  - "RAS" requirements are needed.

Yokohama Research Lab.
Linux Technology Center

0-2 RAS

HITACHI
Inspire the Next

- Reliability
  - To identify problems before release
    e.g. Bug fixing, Testing
- Availability
  - To continue the operation even if a problem occurs
    e.g. HA cluster system
- Serviceability
  - To find out the root cause of the problem certainly in order to be able to solve it permanently
    e.g. Logging, Tracing, Memory dump

- Do the systems satisfy these requirements in current upstream kernel?
  - Will talk about 'R' and 'S'

Yokohama Research Lab.
Linux Technology Center

Activities

1. Fix a deadlock problem on NMI dump（R）
2. Improve data reception latency on serial devices（R）
3. Save names of more processes in ftrace（S）
4. Solve the printk message fragmentation problem（S）

Yokohama Research Lab.
Linux Technology Center

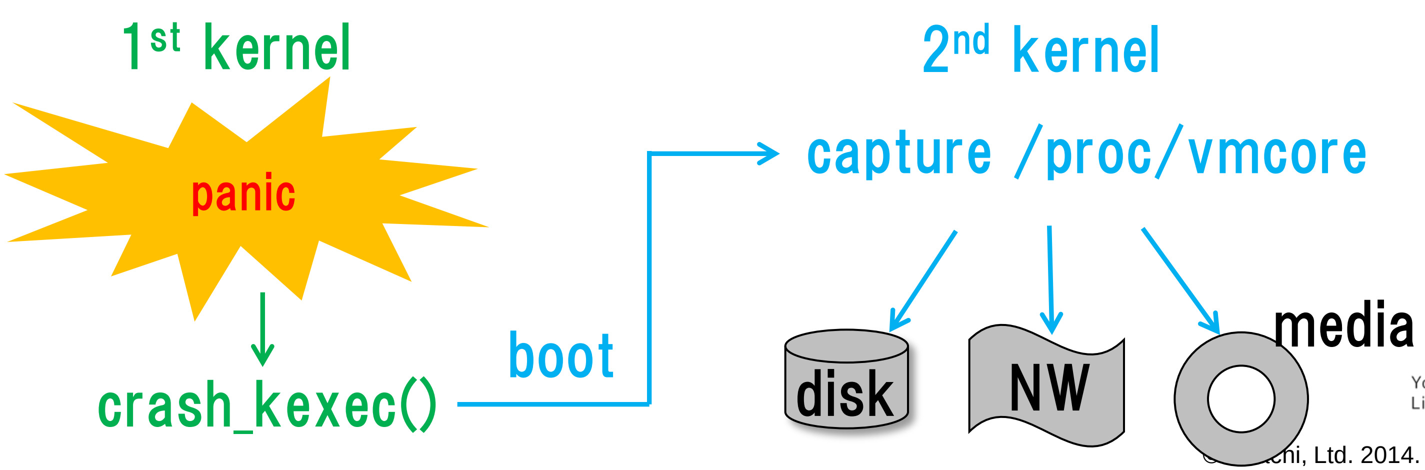
Activities

1. Fix a deadlock problem on NMI dump（R）
2. Improve data reception latency on serial devices（R）
3. Save names of more processes in ftrace（S）
4. Solve the printk message fragmentation problem（S）

Yokohama Research Lab.
Linux Technology Center

1-1 Memory dump deadlock – introduction

- We get memory dump via Kdump when serious problems, which induce panic or oops, occur.
- Kdump
  - Kernel crash dumping feature based on Kexec
    1. In 1st kernel , kernel panic occurs.
    2. Execute crash_kexec() in panic() and save the memory
    3. Boot 2nd kernel (capture kernel) and copy /proc/vmcore
- When kernel panic occurs via NMI, Kdump operation sometimes stops before booting 2nd kernel.



1-2 Memory dump deadlock - reason

HITACHI Inspire the Next

- The cause of the stop is deadlock on ioapic_lock in NMI context.
  - panic()->crash_kexec()->…->disable_IOAPIC() -> …->ioapic_read_entry()

```
ioapic_read_entry()
{
        raw_spin_lock_irqsave(&ioapic_lock, flags);
        eu.entry = __ioapic_read_entry(apic, pin);
        raw_spin_unlock_irqstore(&ioapic_lock, flags);
}
```

- The scenario is …
  1. Get ioapic_lock for rebalancing IRQ (irq_set_affinity)
  2. Inject NMI while locking ioapic_lock
  3. Panic caused by NMI occurs
  4. Try to execute Kdump
  5. Deadlock in ioapic_read_entry()

Yokohama Research Lab.
Linux Technology Center

1-3 Memory dump deadlock - fixing

HITACHI
Inspire the Next

- Fixed this problem by initializing ioapic_lock before disable_IO_APIC():

```
native_machine_crash_shutdown()
{
            …
#ifdef CONFIG_X86_IO_APIC
+           /* Prevent crash_kexec() from deadlocking on ioapic_lock. */
+           ioapic_zap_locks();
            disable_IO_APIC();
#endif
            …
}
```

- This problem has been already fixed in current kernel.
  (from kernel-3.11)

Activities

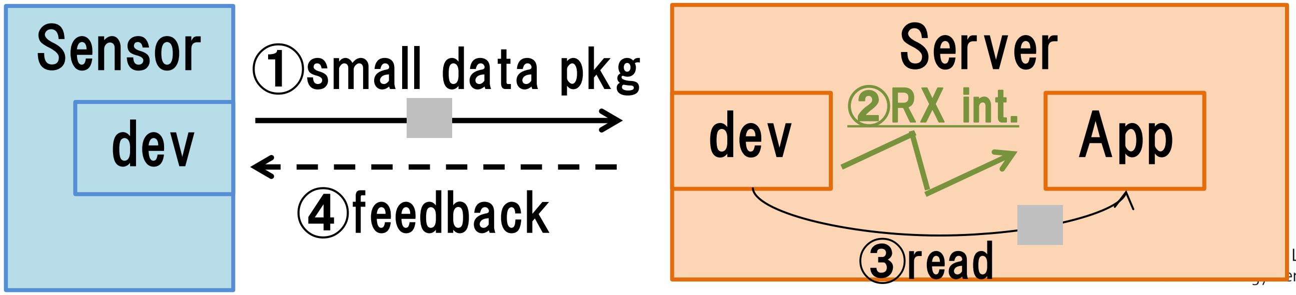
1. Fix a deadlock problem on NMI dump (R)
2. **Improve data reception latency on serial devices (R)**
3. Save names of more processes in ftrace (S)
4. Solve the printk message fragmentation problem (S)

Yokohama Research Lab.
Linux Technology Center

2-1 Serial RX interrupt frequency – introduction

- Serial devices are mainly used not only for embedded systems but also for mission-critical systems.
  - Maintenance
  - Sensor feedback
    - Serial communication has the specialty that it is resistant for noise.
    - For a control system(one of mission-critical systems), long-distance communication is needed.

- If we have a sensor which sends small data packages each time and must control a device based on the sensor feedback, the RX interrupt should be triggered for each packages.



2-2 Serial RX interrupt frequency - problem

- A test requirement of a system was that the serial communication time between send and receive has to be within *3msec*.
- When we measured the time on 16550A, it took *10msec*.
  - It did not change even if the receiver application was operated as a real time application.
  - We analyzed this by using event tracer of ftrace.
  - Hard IRQ of the serial device interrupts once each 10msec, so this is caused by a HW specification or the device driver.

```
                timestamp[sec]                                    ~10msec
<idle>-0 [001] 2689.160668: irq_handler_entry: irq=4 name=serial
<idle>-0 [001] 2689.170653: irq_handler_entry: irq=4 name=serial
<idle>-0 [001] 2689.180634: irq_handler_entry: irq=4 name=serial
<idle>-0 [001] 2689.190620: irq_handler_entry: irq=4 name=serial
```

2-3 Serial RX interrupt frequency - reason

- HW spec of 16550A
  - 16bytes FIFO buffer
  - Flow Control Register(FCR)
    - 2bit register
    - Changeable RX interrupt trigger of 1, 4, 8, or 14 bytes for the FIFO buffer
      (0b00=1byte, 0b01=4bytes, 0b10=8bytes, 0b11=14bytes)
- In Linux, the trigger is hard-coded as 8bytes.

```
[PORT_16550A] = {
        .name = "16550A",
        .fifo_size = 16,
        .tx_loadsz = 16,
        .fcr = UART_FCR_ENABLE_FIFO | UART_FCR_R_TRIG_10,
        .flags = UART_CAP_FIFO,
},
```

8bytes trigger

  - For 9600baud, an interrupt per 10msec is consistent.

(start + octet + stop * 2 + parity) / 9600(baud) = 1/800 (sec/byte) = 1.25(msec/byte)

start: 1bit, octet: 8bit, stop: 1bit * 2, parity: 1bit

1.25(msec/byte) * 8(byte) = **10msec**

2-4 Serial RX interrupt frequency - temporary fixing

HITACHI
Inspire the Next

- Changed FCR as a test

```
[PORT_16550A] = {
        .name = "16550A",
        .fifo_size = 16,
        .tx_loadsz = 16,
        .fcr = UART_FCR_ENABLE_FIFO | UART_FCR_R_TRIG_00,
        .flags = UART_CAP_FIFO,
},
```

1byte trigger

- Result
  - The interrupt frequency is once each 1.25msec.

timestamp[sec]

1.25msec

```
<idle>-0 [001] 3216.436959: irq_handler_entry: irq=4 name=serial
<idle>-0 [001] 3216.438209: irq_handler_entry: irq=4 name=serial
<idle>-0 [001] 3216.439454: irq_handler_entry: irq=4 name=serial
<idle>-0 [001] 3216.440706: irq_handler_entry: irq=4 name=serial
```

- We need a configurable RX interrupt trigger.

Yokohama Research Lab.
Linux Technology Center

2-5 Serial RX interrupt frequency – tunable patch

HITACHI Inspire the Next

- Added new I/F to the serial driver
  - Tunable RX interrupt trigger frequency
    - High frequency(1byte trigger) → low latency
    - Low frequency(14byte trigger) → low CPU overhead
  - Usability problems of 1st/2nd patch:
    - ✓ Using ioctl(2) (c.f. using echo command is better)
    - ✓ Interrupt frequency can be changed only **after** opening serial fd
    - ✓ Cannot read FCR value (FCR is a write-only register)
  - Change the ioctl(2) to sysfs I/F after discussion in a Linux community
    - Set the interrupt trigger byte (if val is invalid, nearest lower val is set.)

```
# echo  1  > rx_trig_byte                    /* 1byte trigger*/
```

    - User can read/write the trigger any time.
      - The driver keeps FCR value if user changes interrupt trigger.

- This new feature will be able to be used from kernel-3.17.

Yokohama Research Lab.
Linux Technology Center

HITACHI
Inspire the Next

Activities

1. Fix a deadlock problem on NMI dump (R)
2. Improve data reception latency on serial devices (R)
3. **Save names of more processes in ftrace (S)**
4. Solve the printk message fragmentation problem (S)

Yokohama Research Lab.
Linux Technology Center

3-1 PID-process name table in ftrace - introduction

HITACHI
Inspire the Next

- ftrace is in-kernel tracer for debugging and analyzing problems.
- ftrace records PIDs and process names in order to specify process context of an operation.
  - If process name is indicated in a trace result, a user can understand who executed the program by doing grep with the process name.

- In the trace file, process names are sometimes output as <…>:

name-PID

```
<...>-2625 [002] .... 209630.888186: sys_write(fd: 8, buf: 7fd0ef836968, count: 8)
<...>-2625 [002] .... 209630.888186: sys_enter: NR 1 (8, 7fd0ef836968, 8, 20, 0, a41)
<...>-2625 [002] .... 209630.888186: kfree: call_site=ffffffff810e410c ptr= (null)
<...>-2625 [002] .... 209630.888187: sys_write -> 0x8
```

3-2 PID-process name table in ftrace – reason

- ftrace has saved_cmdlines file storing the PID-process name mapping table.
  - It stores the list of **128** processes that hit a tracepoint.

```
# cat saved_cmdlines
13 ksoftirqd/1
10009 python
1718 gnome-panel
500 jbd2/sda5-8
…
```

  - If the number of processes that hit a tracepoint exceeds 128, the oldest process name is overwritten.
  - How does ftrace manage this table?

3-3 PID-process name table in ftrace - current

HITACHI
Inspire the Next

- Read trace file (get process name from PID)

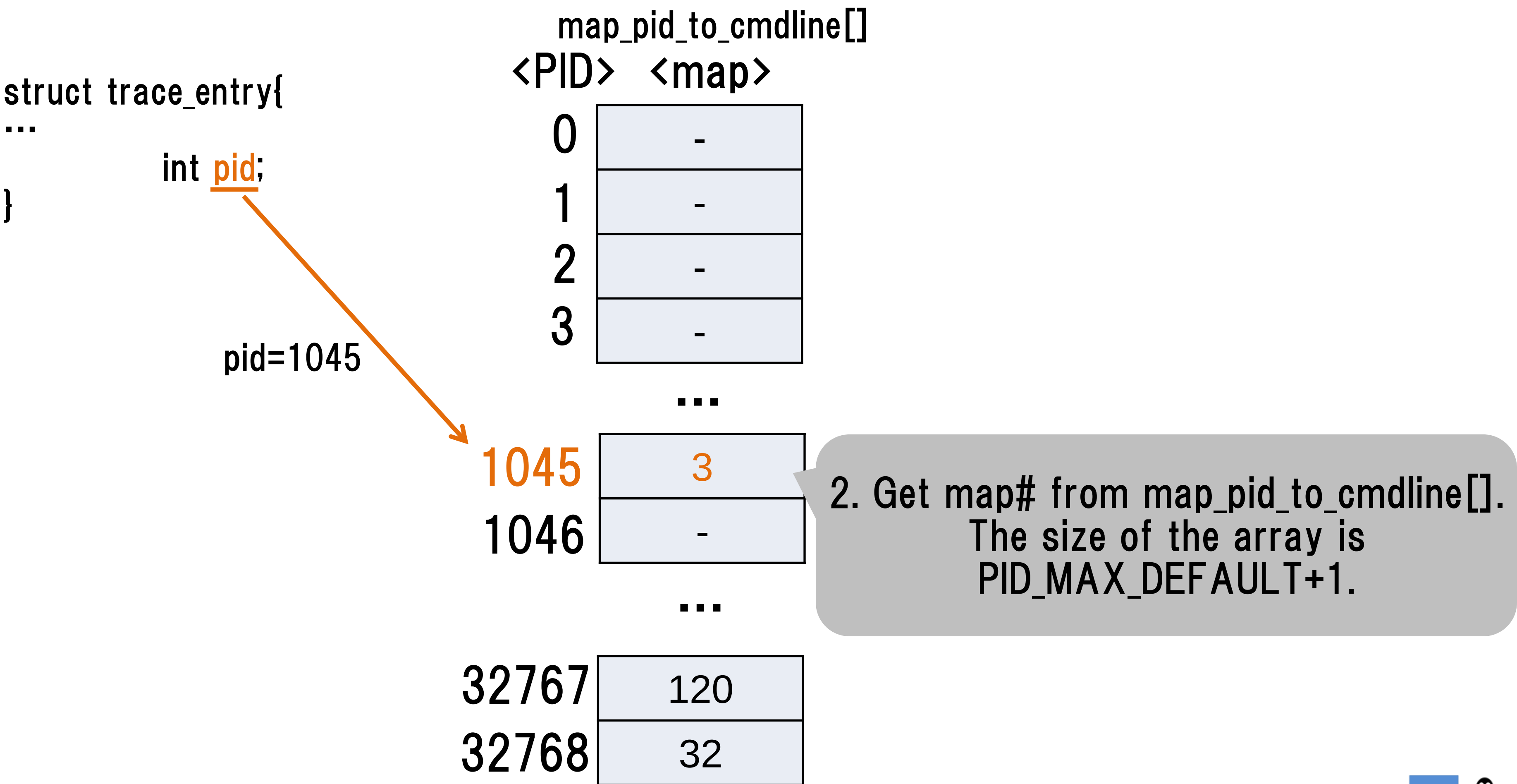
```
struct trace_entry{
…
            int pid;
}
```

1. Get a pid member in trace_entry structure of each trace event

Yokohama Research Lab.
Linux Technology Center

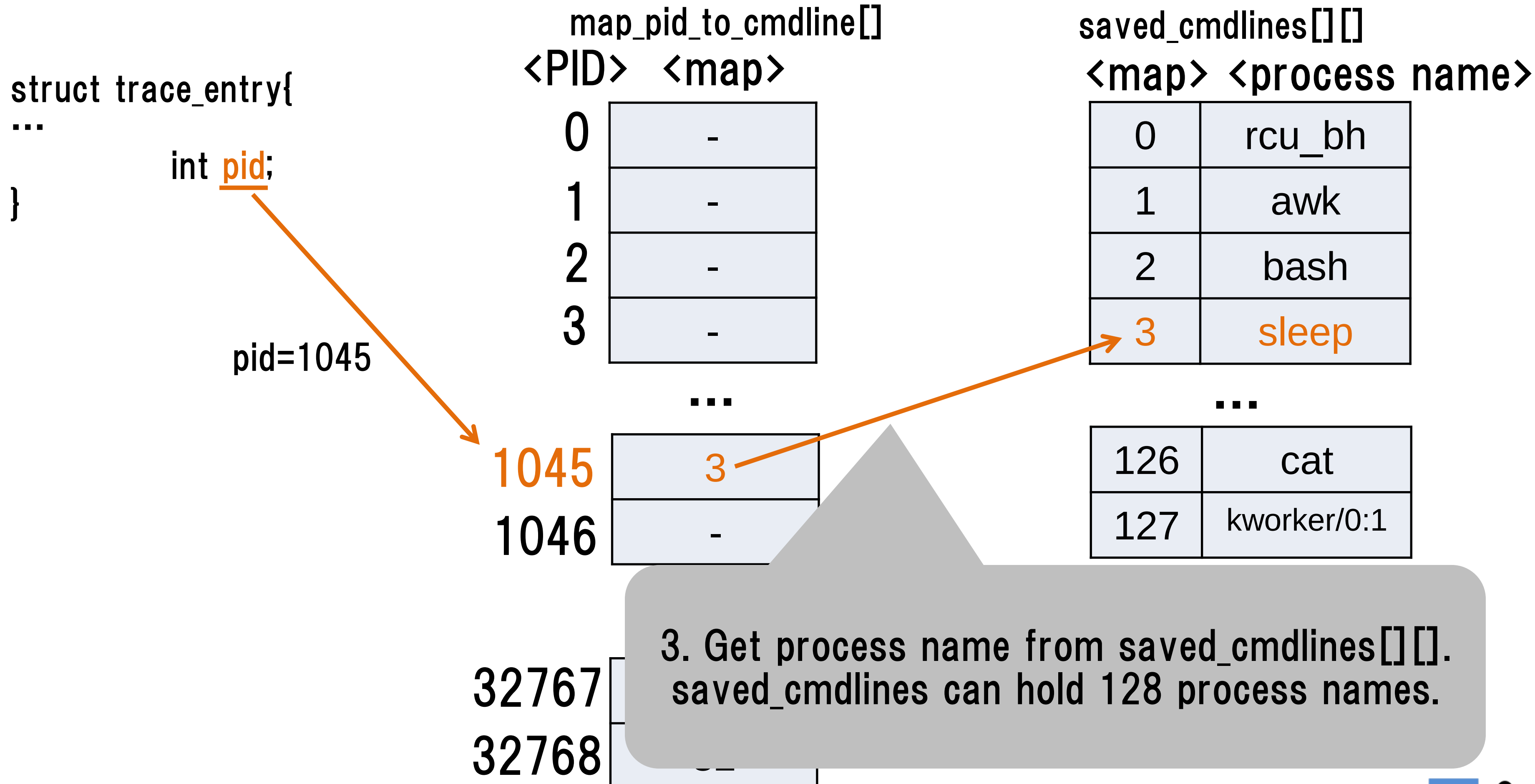
3-4 PID-process name table in ftrace - current

- Read trace file (get process name from PID)



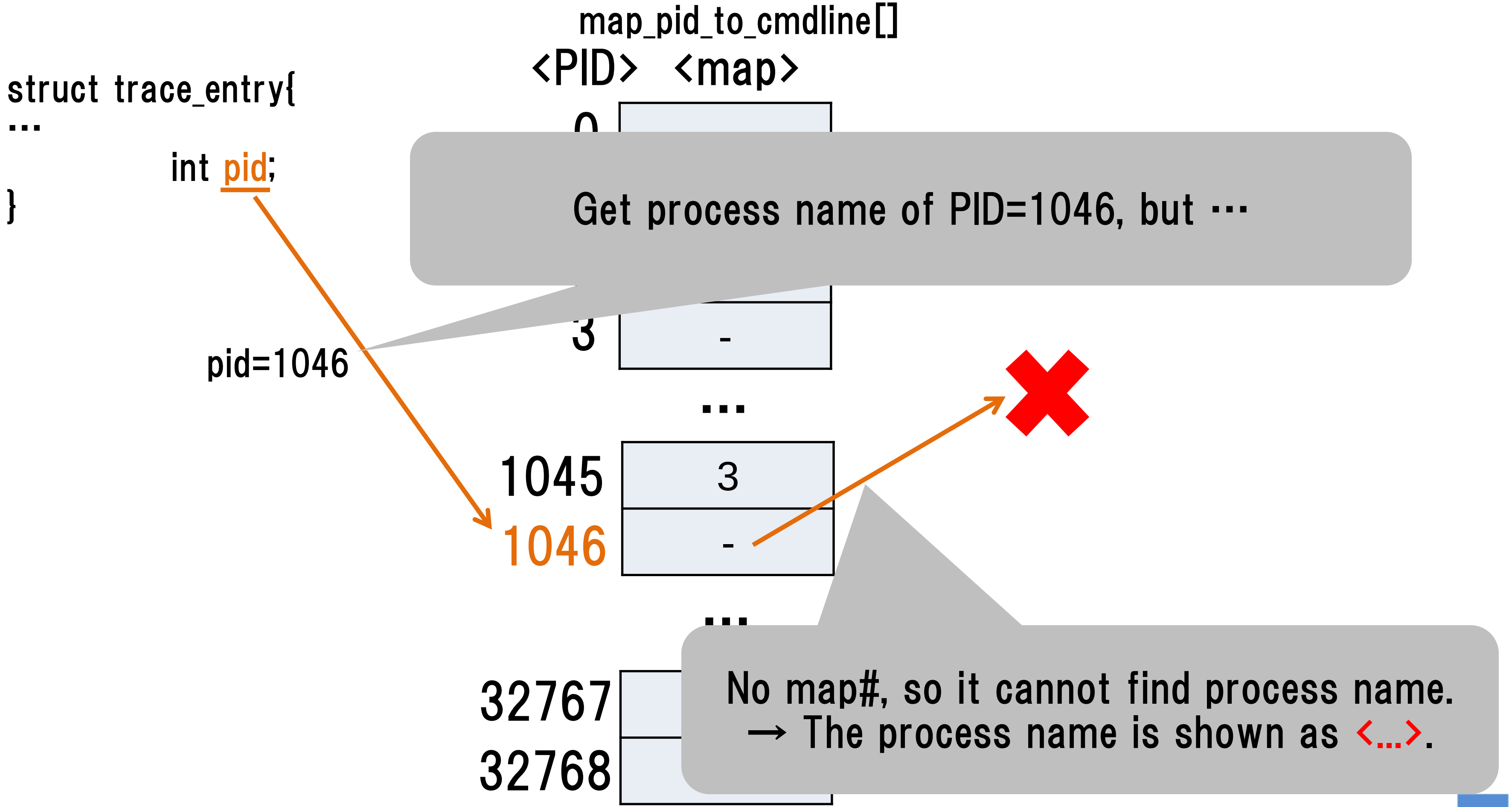
3-5 PID-process name table in ftrace - current

- Read trace file (get process name from PID)



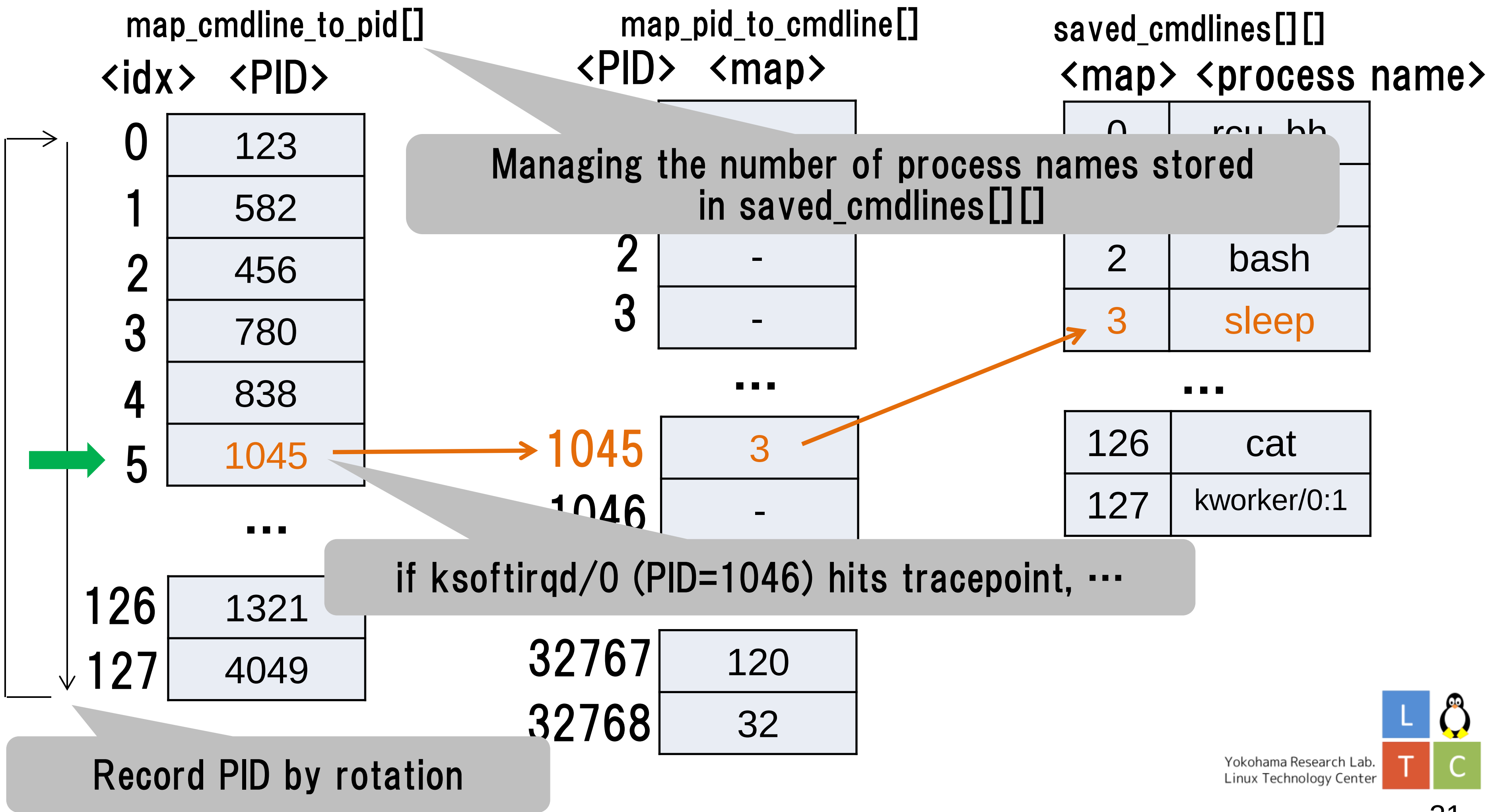
3-6 PID-process name table in ftrace – current

- Read trace file (get process name from PID)



3-7 PID-process name table in ftrace – current

- Store map information in saved_cmdlines

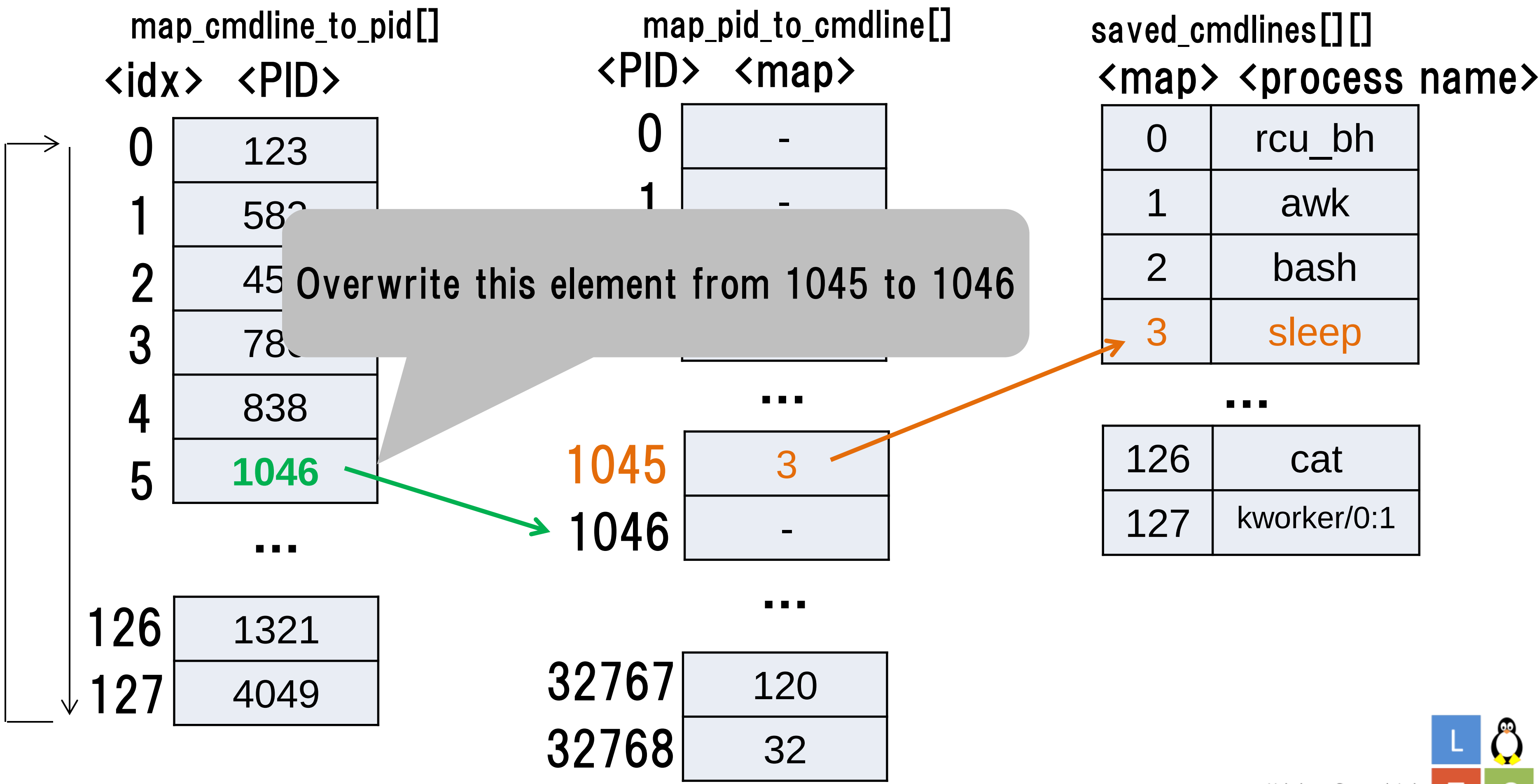


Yokohama Research Lab.
Linux Technology Center

3-8 PID-process name table in ftrace – current

HITACHI
Inspire the Next

- Store map information in saved_cmdlines

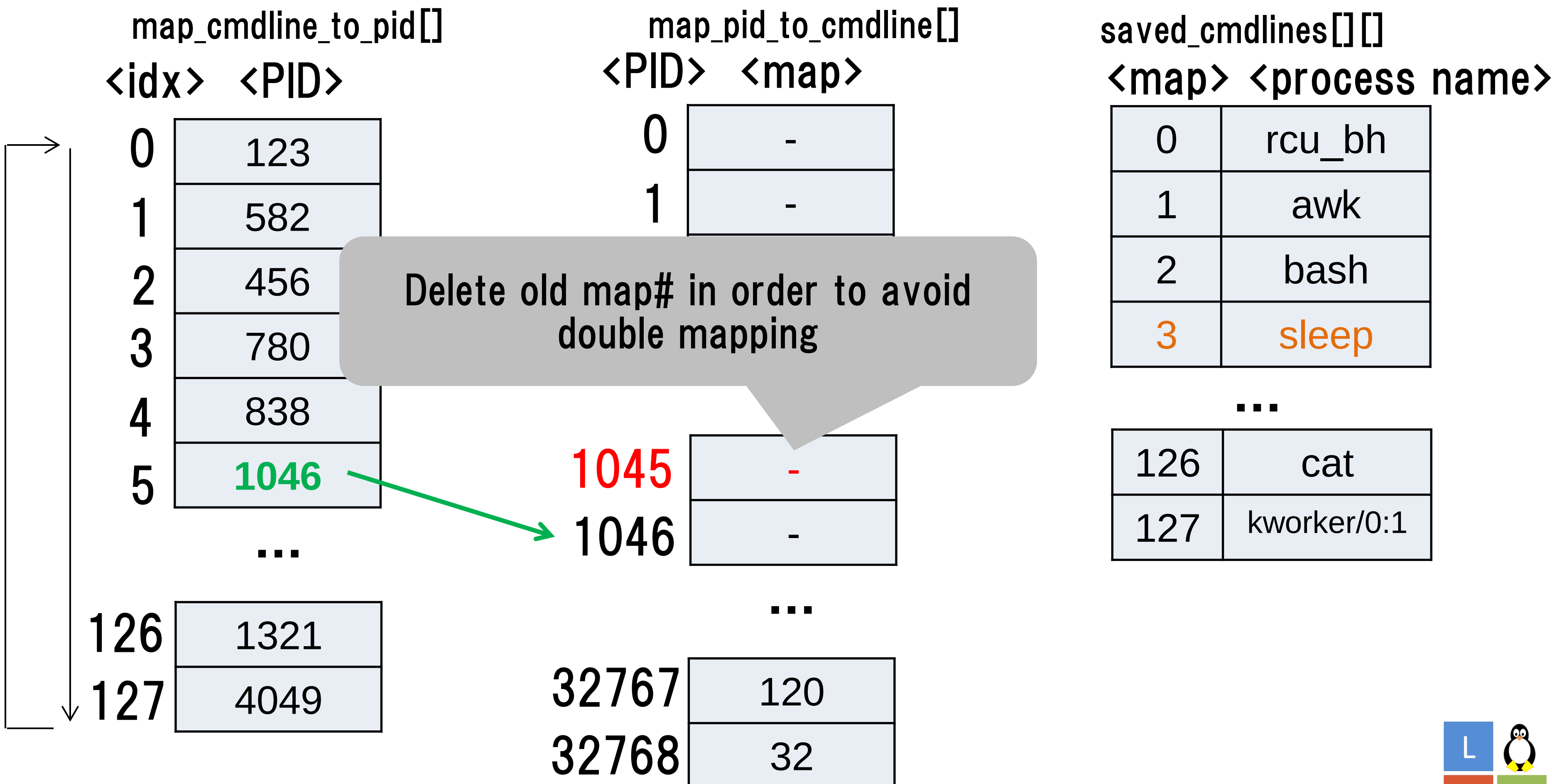


Yokohama Research Lab.
Linux Technology Center

3-9 PID-process name table in ftrace - current

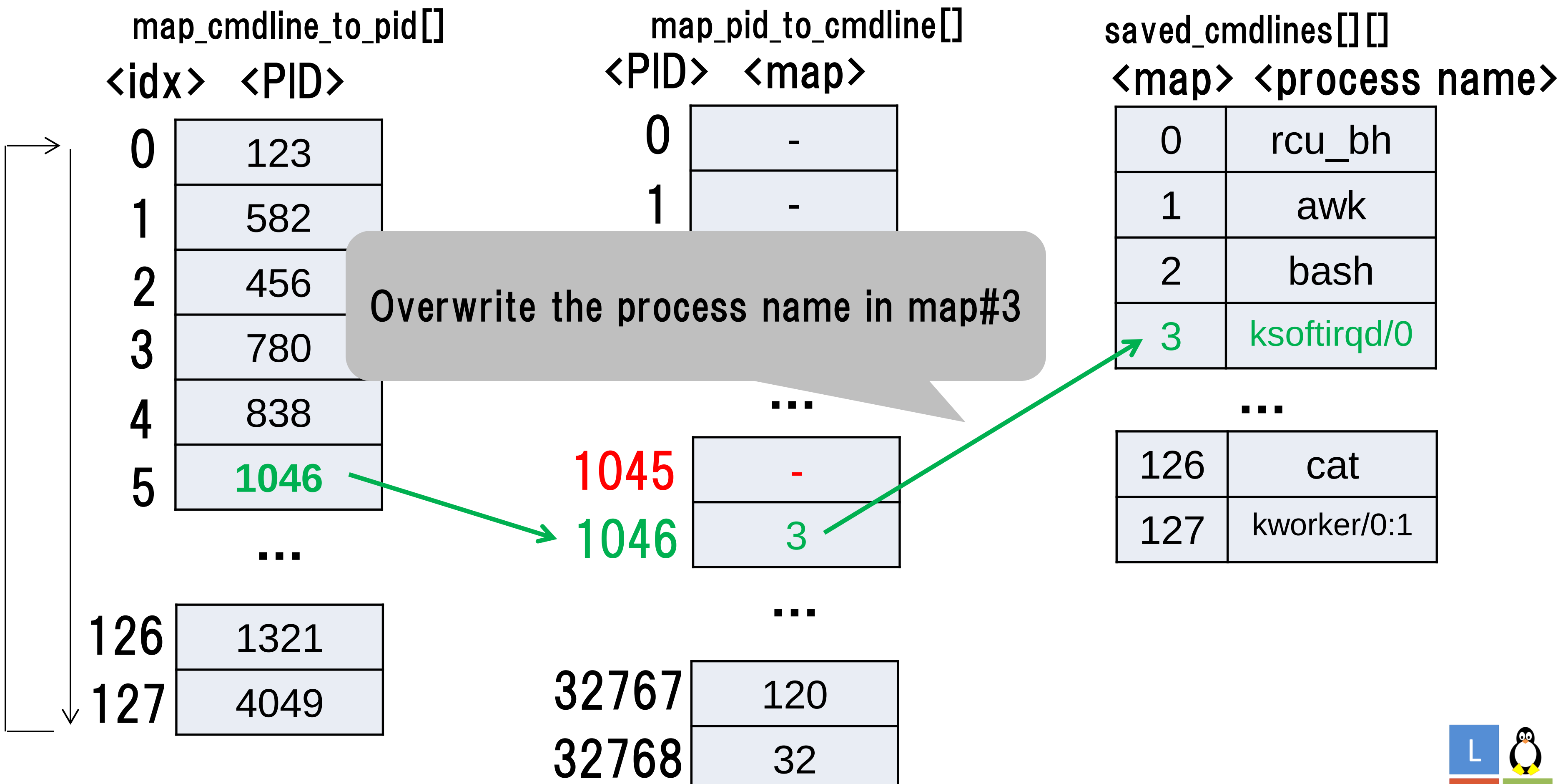
HITACHI
Inspire the Next

- Store map information in saved_cmdlines



3-10 PID-process name table in ftrace - current

- Store map information in saved_cmdlines



3-10 PID-process name table in ftrace - current

- Store map information in saved_cmdlines

map_cmdline_to_pid[]

<idx>	<PID>
0	123
1	582
2	456
3	780
4	838
5	1046
...	...
126	1321
127	4049

map_pid_to_cmdline[]

<PID>	<map>
0	-
1	-
2	-
3	-
...	...
1045	-
1046	3
...	...

saved_cmdlines[][]

<map>	<process name>
0	rcu_bh
1	awk
2	bash
3	ksoftirqd/0
...	...
126	cat
127	kworker/0:1

Develop tunable I/F

3-11 PID-process name table in ftrace – tunable patch

HITACHI Inspire the Next

- We added the changeable I/F 'saved_cmdlines_size' to expand the max number of saved process names.
  - Read/write saved_cmdlines_size
    - For write, all saved_cmdlines information are cleared.
  - Max size: PID_MAX_DEFAULT(32768)
    - If we set 32768, all process names can be stored.

```
# cat saved_cmdlines_size
  128      /* defalut value*/

// Switch to new saved_cmdlines buffers
# echo  1024  > saved_cmdlines_size

# cat saved_cmdlines_size
  1024    /* Store 1024 process names */
```

- This new feature can be used from kernel-3.16.

Yokohama Research Lab.
Linux Technology Center

Activities

1. Fix a deadlock problem on NMI dump（R）
2. Improve data reception latency on serial devices（R）
3. Save names of more processes in ftrace（S）
4. **Solve the printk message fragmentation problem（S）**

Yokohama Research Lab.
Linux Technology Center

4-1 printk fragmentation problem – introduction

- printk message outputs error logging or debugging information in kernel.
  - We handle automatically printk messages in user space in order to detect that the system has became unstable.
  - We want the kernel to output printk as expected.
  - printk messages are sometimes **mixed with similar messages**.
    - It is difficult to automatically handle an event from mixed messages.
- **mixed** kernel error messages in SCSI layer

Which process does this message belong to?

```
[110781.736171] sd 2:0:0:0: [sdb]
[110781.736170] sd 3:0:0:0: [sdc] Unhandled sense code
[110781.736172] sd 3:0:0:0: [sdc]
[110781.736175] Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE
[110781.736177] sd 3:0:0:0: [sdc]
[110781.736178] Sense Key : Medium Error [current]
[110781.736187] Sense Key : Recovered Error
[110781.736189] [current]
```

4-2 printk fragmentation problem – introduction

HITACHI
Inspire the Next

- Mixed messages can occur when multiple printk() are executed at the same time.

<CPU0>

```
printk("sd 2:0:0:0: [sdb]¥n");

printk("Sense Key : Medium Error¥n");
```

break into

<CPU1>

```
printk("sd 3:0:0:0: [sdc]¥n");

printk("Sense Key :
        Recovered Error¥n");
```

```
[110781.736171] sd 2:0:0:0: [sdb]
[110781.736177] sd 3:0:0:0: [sdc]
[110781.736178] Sense Key : Medium Error [current]
[110781.736187] Sense Key : Recovered Error
```

Yokohama Research Lab.
Linux Technology Center

4-3 printk fragmentation problem - Solution

- How to solve
  1. Store all continuous messages in local buffer as temporary, and execute printk
     - This idea is rejected by SCSI community.
       – https://lkml.org/lkml/2014/5/20/742
     - To store continuous messages, we need big buffer.
     - This can induce buffer overflow for deep nesting.
     - Of course, memory allocation is invalid.
  2. Add information necessary to merge all fragmented printk messages
     - This idea is also rejected.
       – https://lkml.org/lkml/2014/5/19/285
     - The community said this problem should be fixed for each subsystem.
  3. Use traceevents of ftrace to output atomically only for SCSI layer
     - This is not for all printk messages.

4-4 printk fragmentation problem - Key idea

HITACHI
Inspire the Next

- traceevents can be atomically stored to ring buffer.
  - Kernel preemption is disabled.
  - A ring buffer a CPU

  ➔ We don't need to concern about mixed traceevent.

- Use trace_seq_printf() for traceevent
  - Add event information using not only macros but functions
  - scsi-trace.c has already used this, but it does not have error messages.

  ➔ We added new three traceevents for SCSI error messages.

      - scsi_show_result: output driverbyte, hostbyte
      - scsi_print_sense: output sense key with asc and ascq
      - scsi_print_command: output SCSI command

Yokohama Research Lab.
Linux Technology Center

4-5 printk fragmentation problem – Result

HITACHI Inspire the Next

- A result of dmesg in current kernel

```
[ 6379.535874] sd 2:0:0:0: [sda]
[ 6379.538376] Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE
[ 6379.542083] sd 2:0:0:0: [sda]
[ 6379.544556] Sense Key : Medium Error [current]
[ 6379.549988] sd 2:0:0:0: [sda]
[ 6379.552408] Add. Sense: Unrecovered read error
[ 6379.574040] sd 3:0:0:0: [sdb]
[ 6379.576576] Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE
[ 6379.580299] sd 3:0:0:0: [sdb]
[ 6379.582727] Sense Key : Medium Error [current]
```

- A result of ftrace with our patch

```
scsi_show_result: …[sda] result=(driver=DRIVER_SENSE host=DID_OK)   } atomic
scsi_print_sense: …[sda] Sense Key (Medium Error [current]) \        } atomic
                         Add. Sense (Unrecovered read error)
scsi_show_result: …[sdb] result=(driver=DRIVER_SENSE host=DID_OK)
scsi_print_sense: …[sdb] Sense Key (Medium Error [current]) \
                         Add. Sense (Unrecovered read error)
```

Linux Technology Center

4-6 printk fragmentation problem - current status

HITACHI
Inspire the Next

- Current patch
  - https://lkml.org/lkml/2014/8/8/221
  - Any comments are welcome!

5 Summary

- We are doing community activities for realizing Linux which satisfies RAS requirements for mission-critical systems.
  - Bug fixing (avoid the deadlock on Kdump)
  - Add features (tunable serial RX trigger, tunable saved_cmdlines, and SCSI traceevents)
  - These activities can be used for not only mission-critical systems but also other systems. For example, fragmented printk is a big problem for a support division of system integrators.

Any questions?

HITACHI
Inspire the Next

Yokohama Research Lab.
Linux Technology Center

Legal statements

- Linux is a registered trademark of Linus Torvalds.
- All other trademarks and copyrights are the property of their respective owners.