Proposal of Live Dump

YOSHIDA Masanori
Yokohama Research Laboratory, Hitachi

LinuxCon Japan '12

Agenda

1. What is Live Dump?
2. Implementation
3. Future work

1. What is Live Dump?
2. Implementation
3. Future work

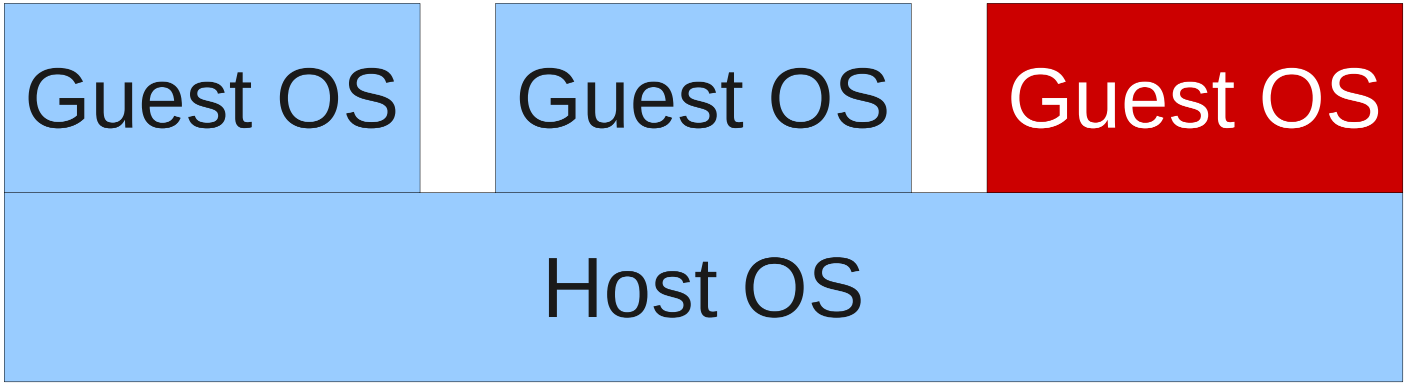
Motivation

- Server consolidation for very important systems

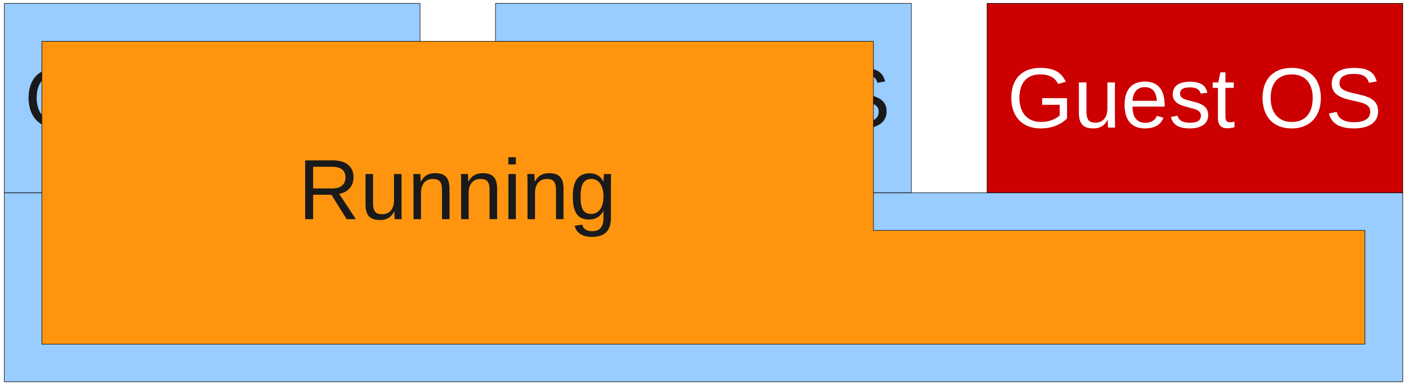
- Problem：Availability vs Serviceability
  - Availability
    - We have to keep a host OS running even after some of guests crash.
  - Serviceability
    - We have to obtain memory dump of both guests and a host to make sure to identify cause of the crash.

Goal

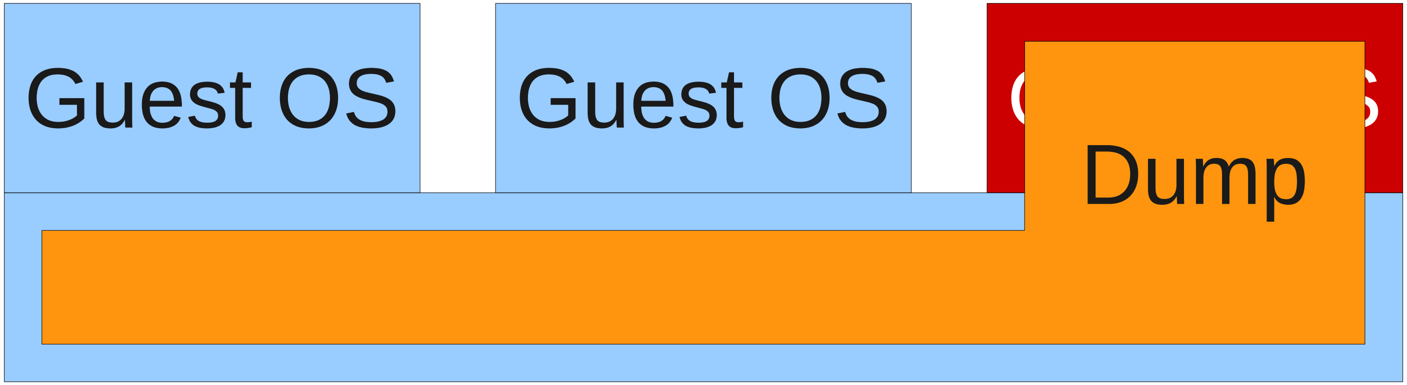
- Assume one guest crashed but others are still running normally.



Goal

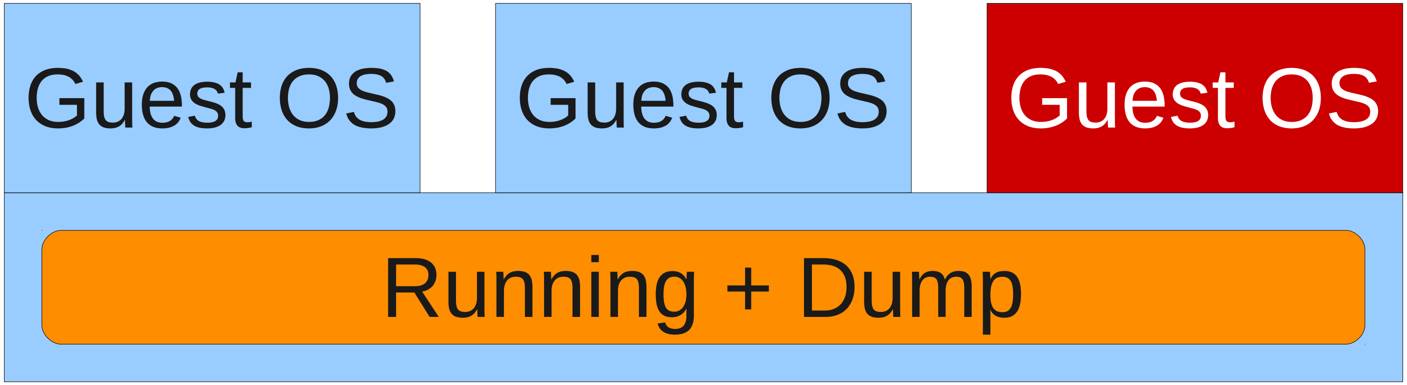


Goal



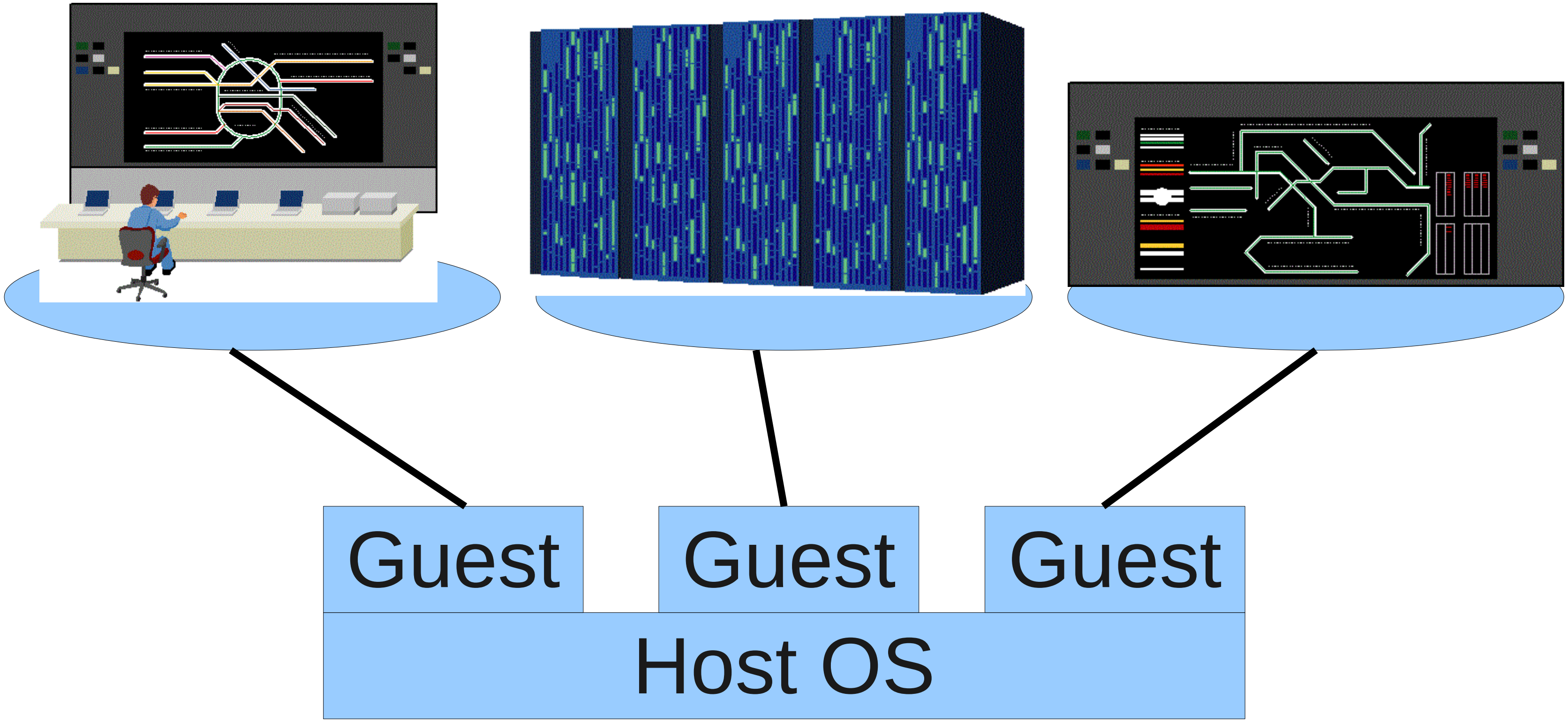
Goal

- Dump consistent memory snapshot of OS,
- without stopping the OS.



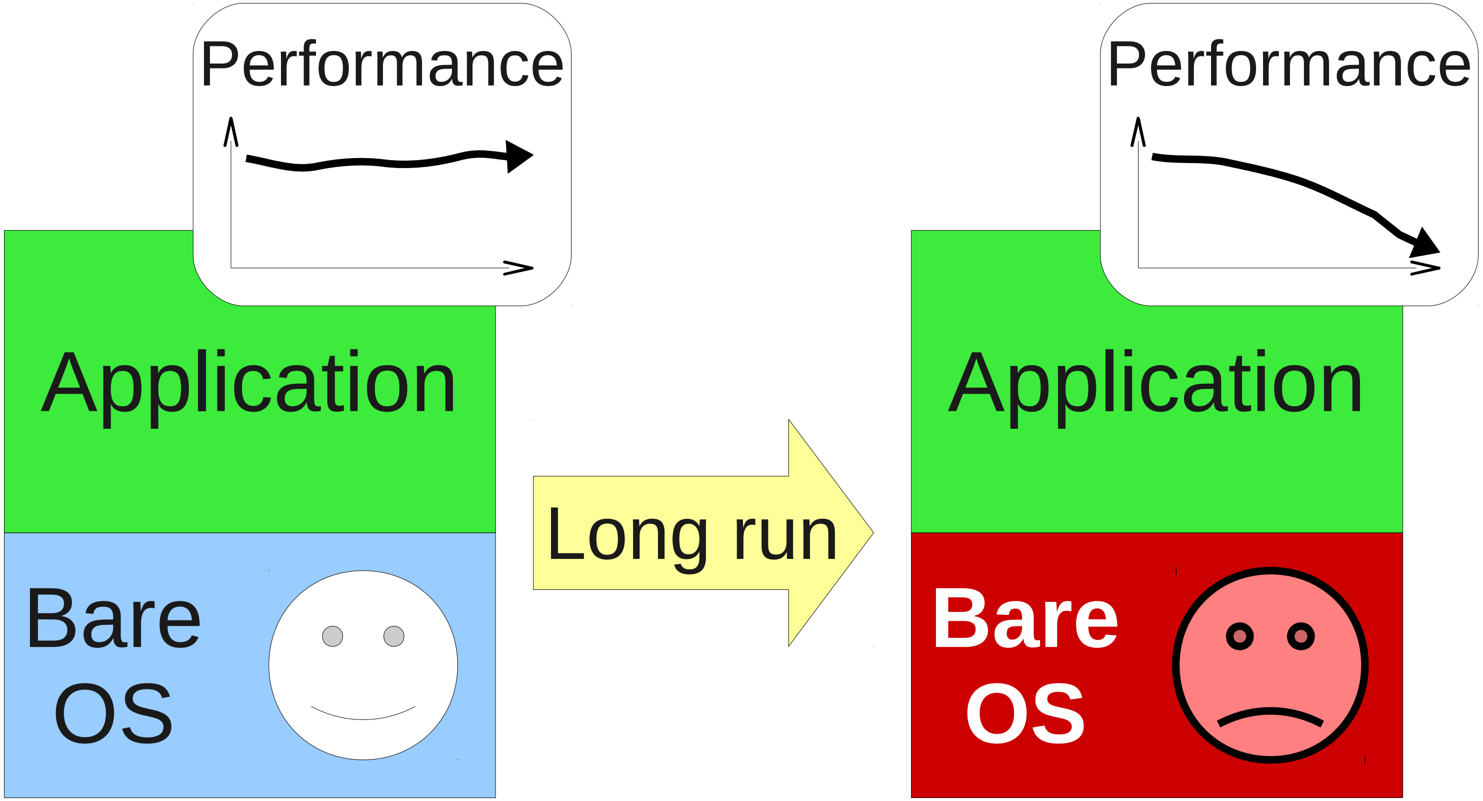
Use case

- Consolidation for very important systems



Use case

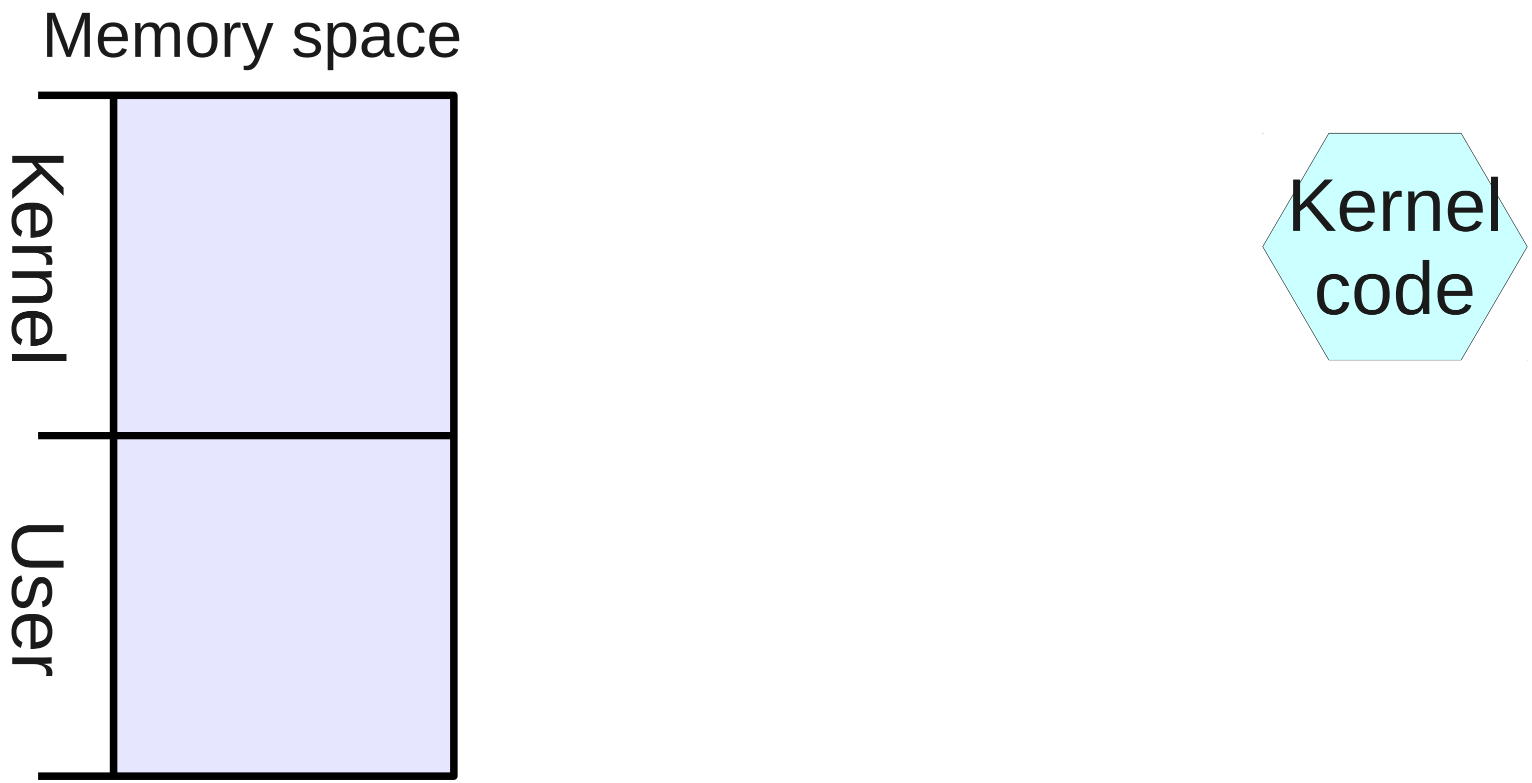
- Non-virtualization case
  - Performance degradation analysis



1. What is Live Dump?
2. Implementation
3. Future work

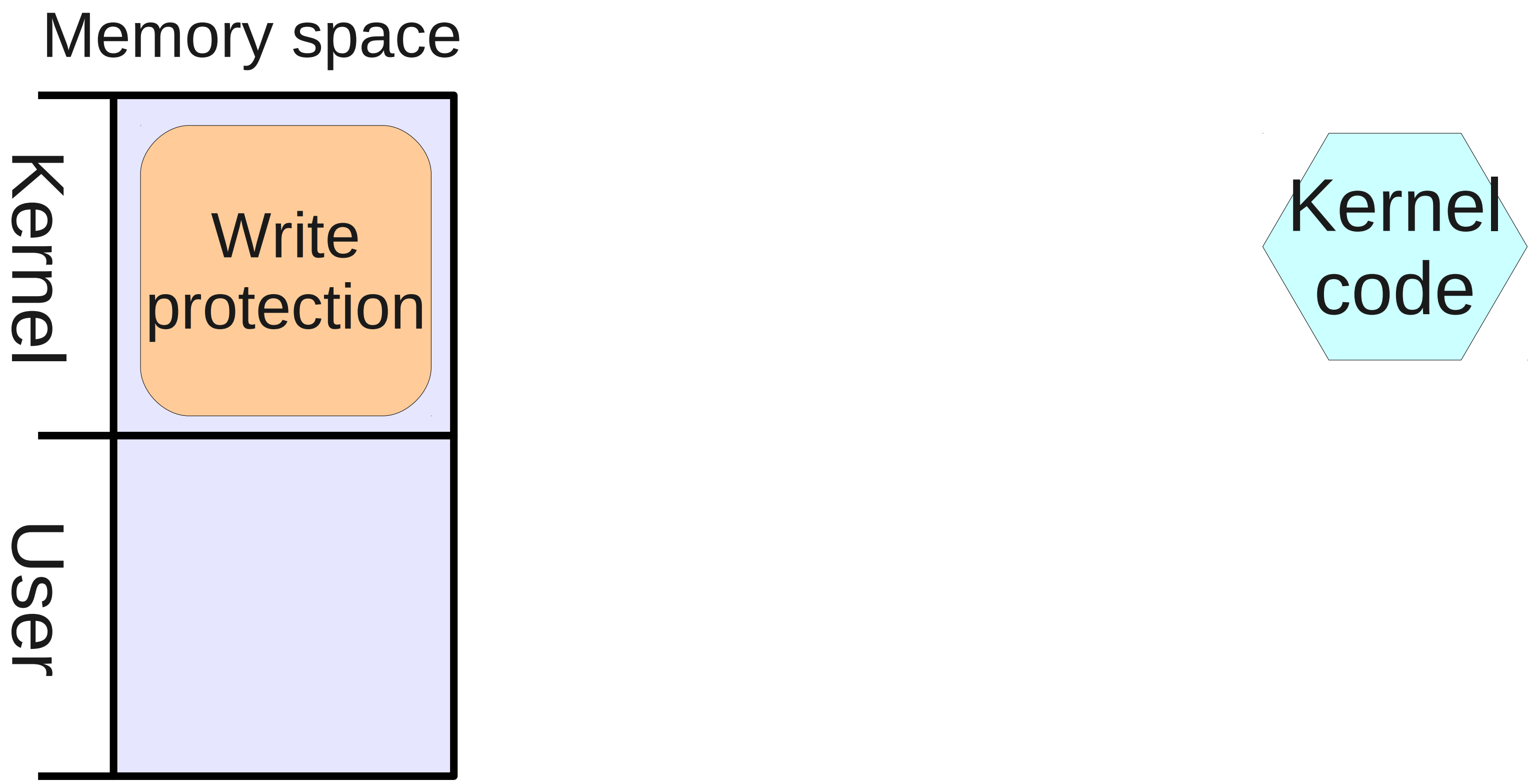
Core idea

- Copy on write



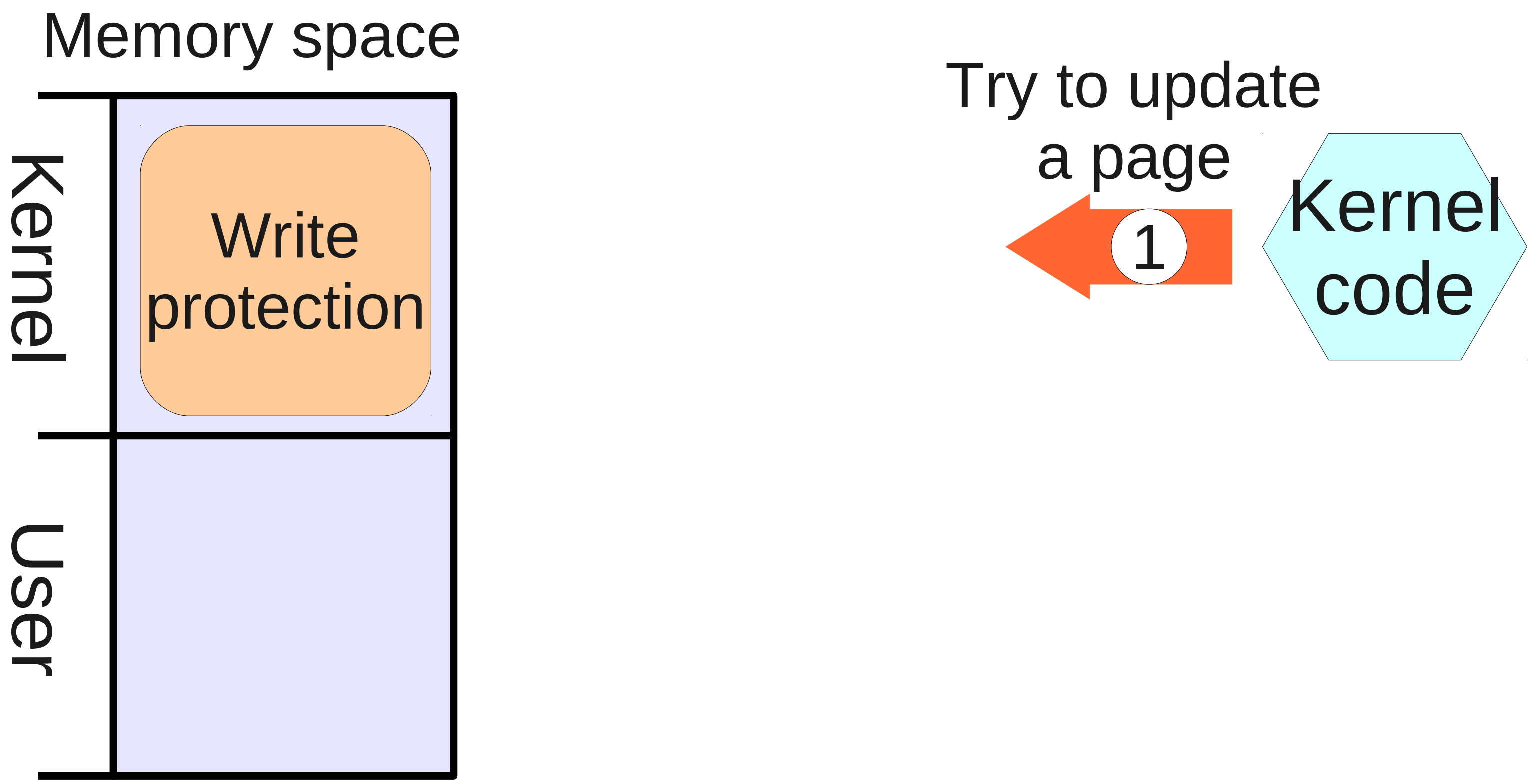
Core idea

- Copy on write



Core idea

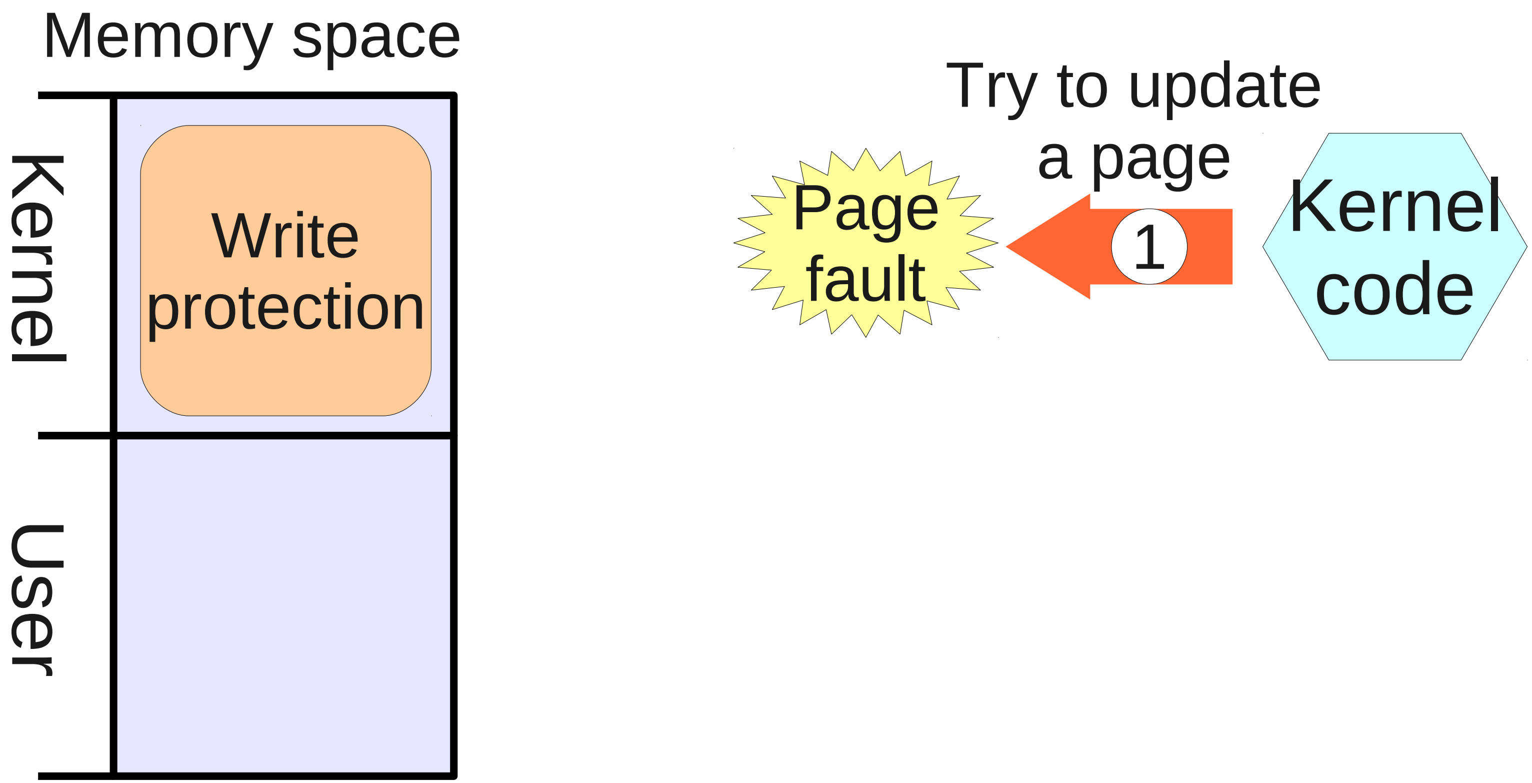
- Copy on write



Copyright (c) 2012 Hitachi LTD., Yokohama Research Lab. All rights reserved.

Core idea

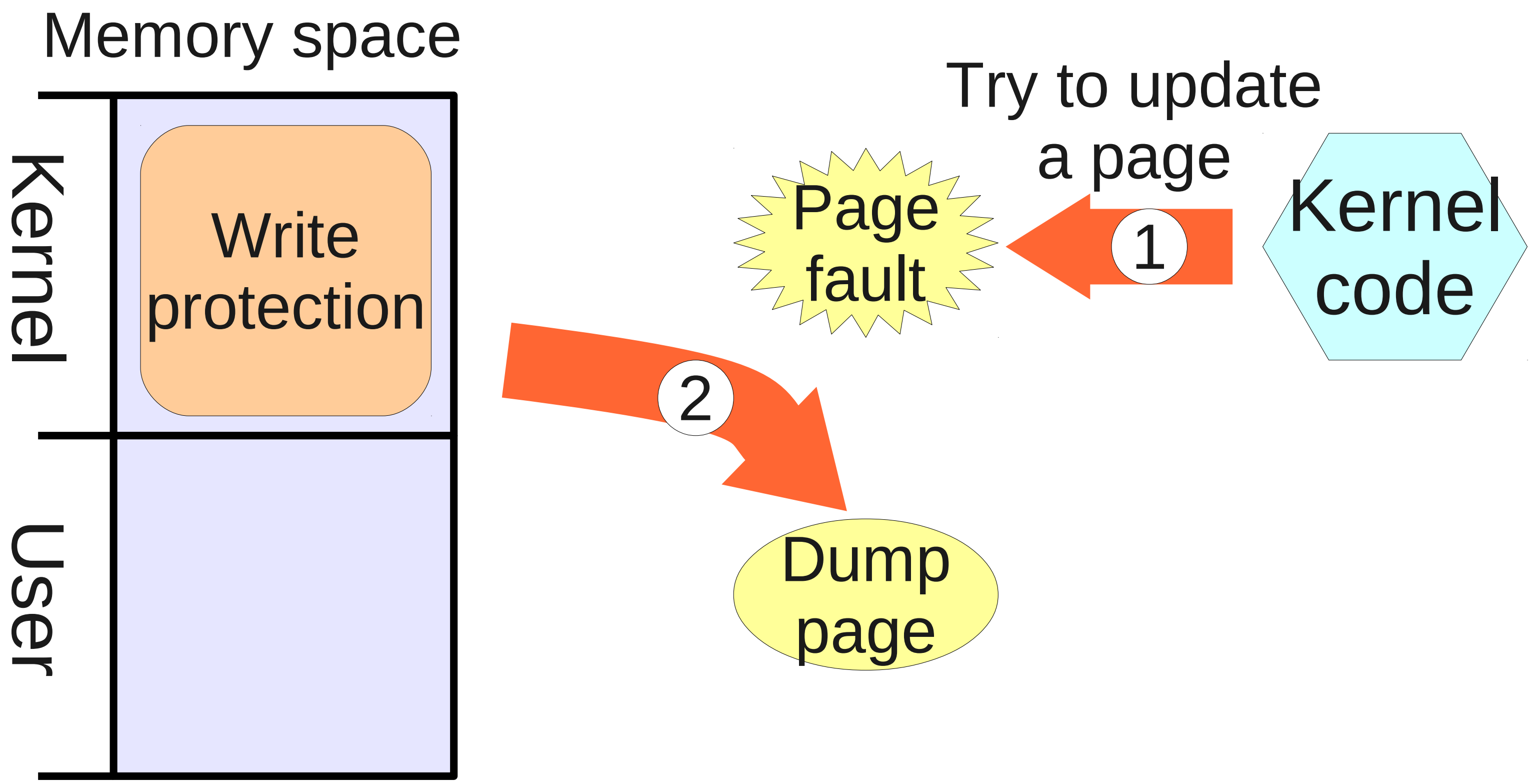
- Copy on write



Copyright (c) 2012 Hitachi LTD., Yokohama Research Lab. All rights reserved.

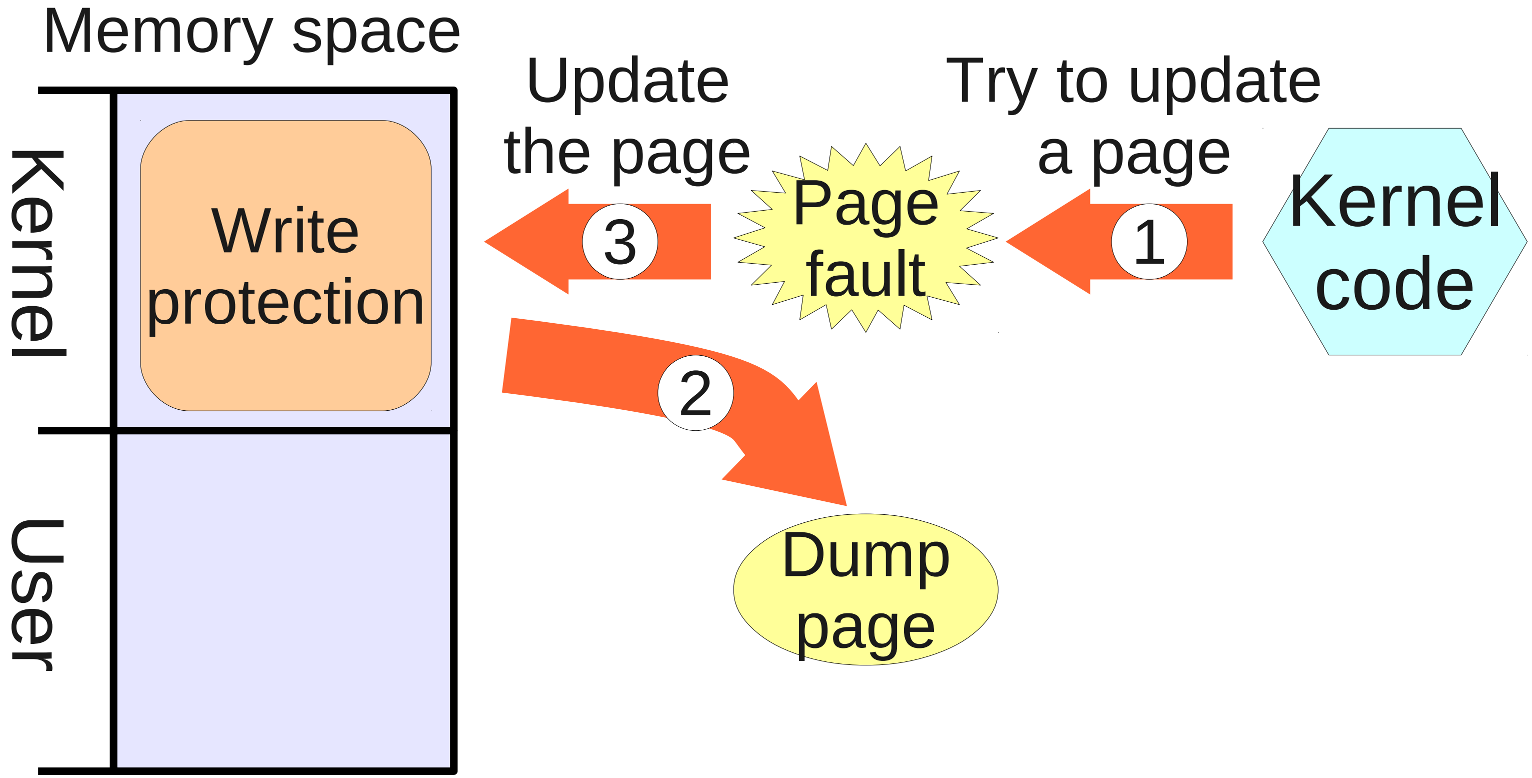
Core idea

- Copy on write



Core idea

- Copy on write



Core idea

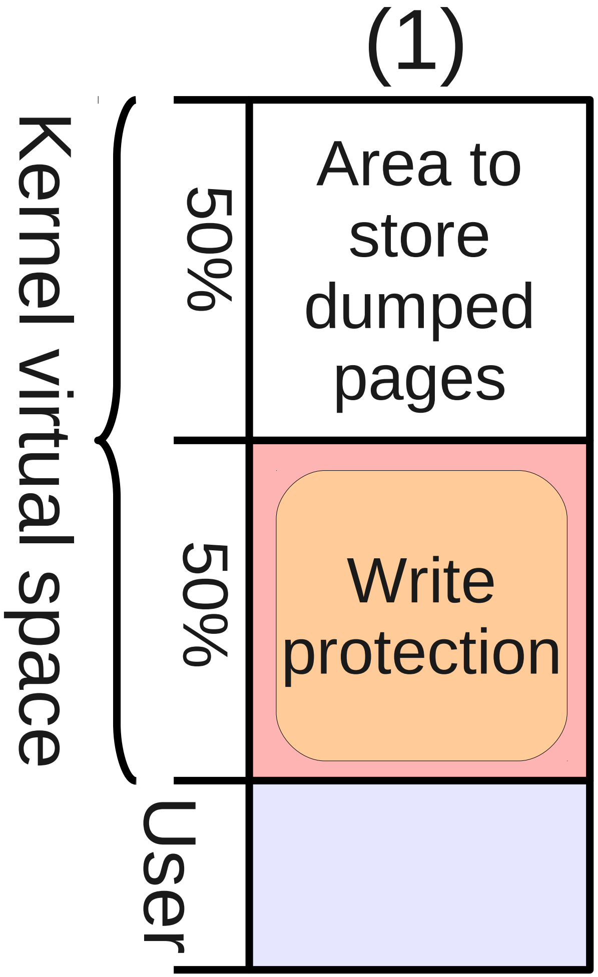
- Copy on write

Limitation

▪Only kernel space is dumped consistently.

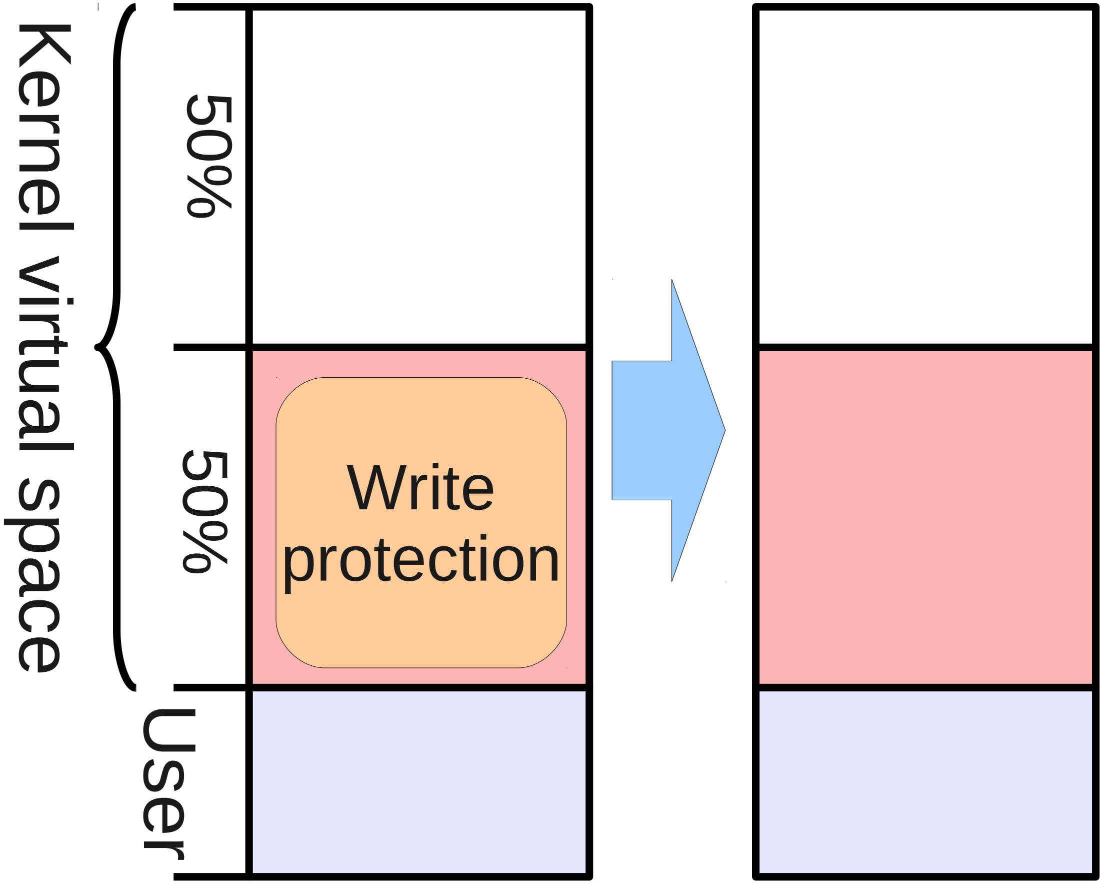
– All phys pages are dumped,
  but those of user space aren't consistent.

Flow of processing



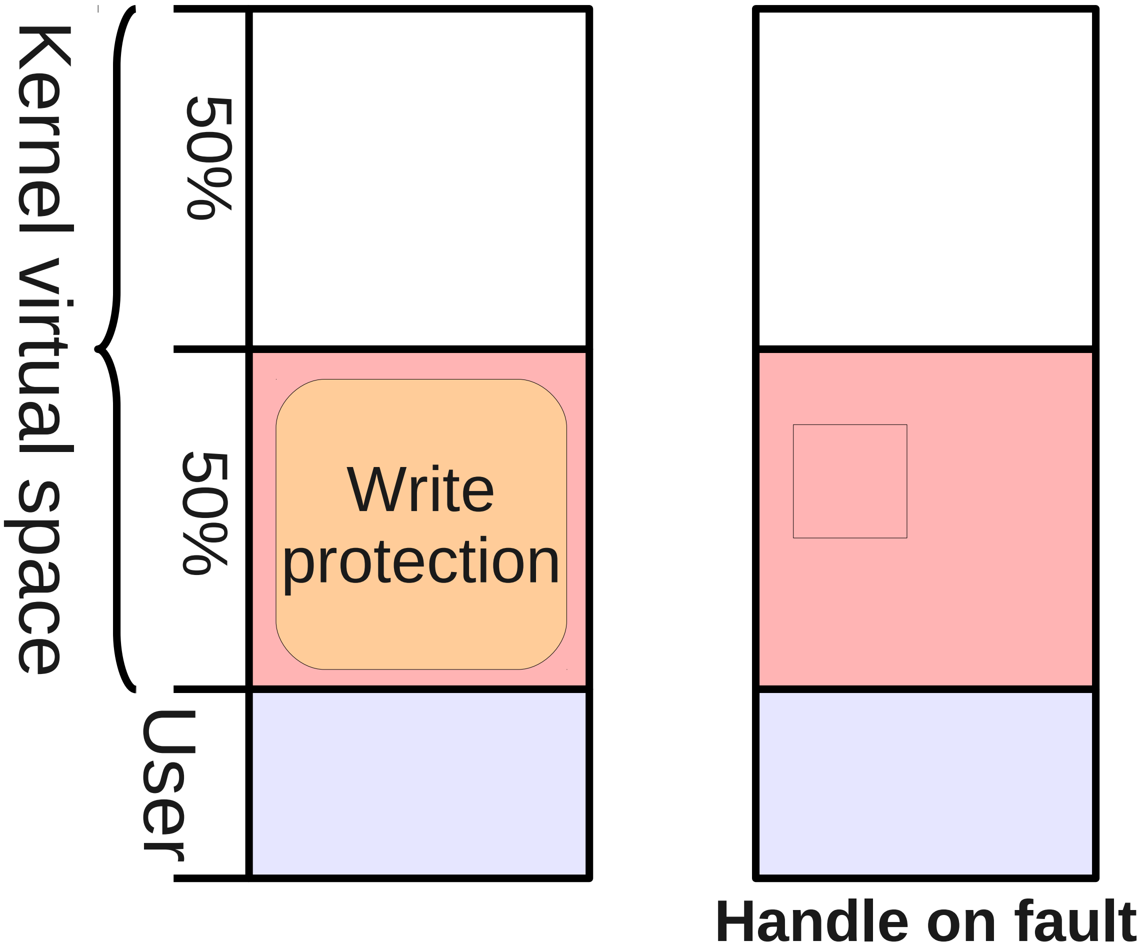
Copyright (c) 2012 Hitachi LTD., Yokohama Research Lab. All rights reserved.

Flow of processing (2)

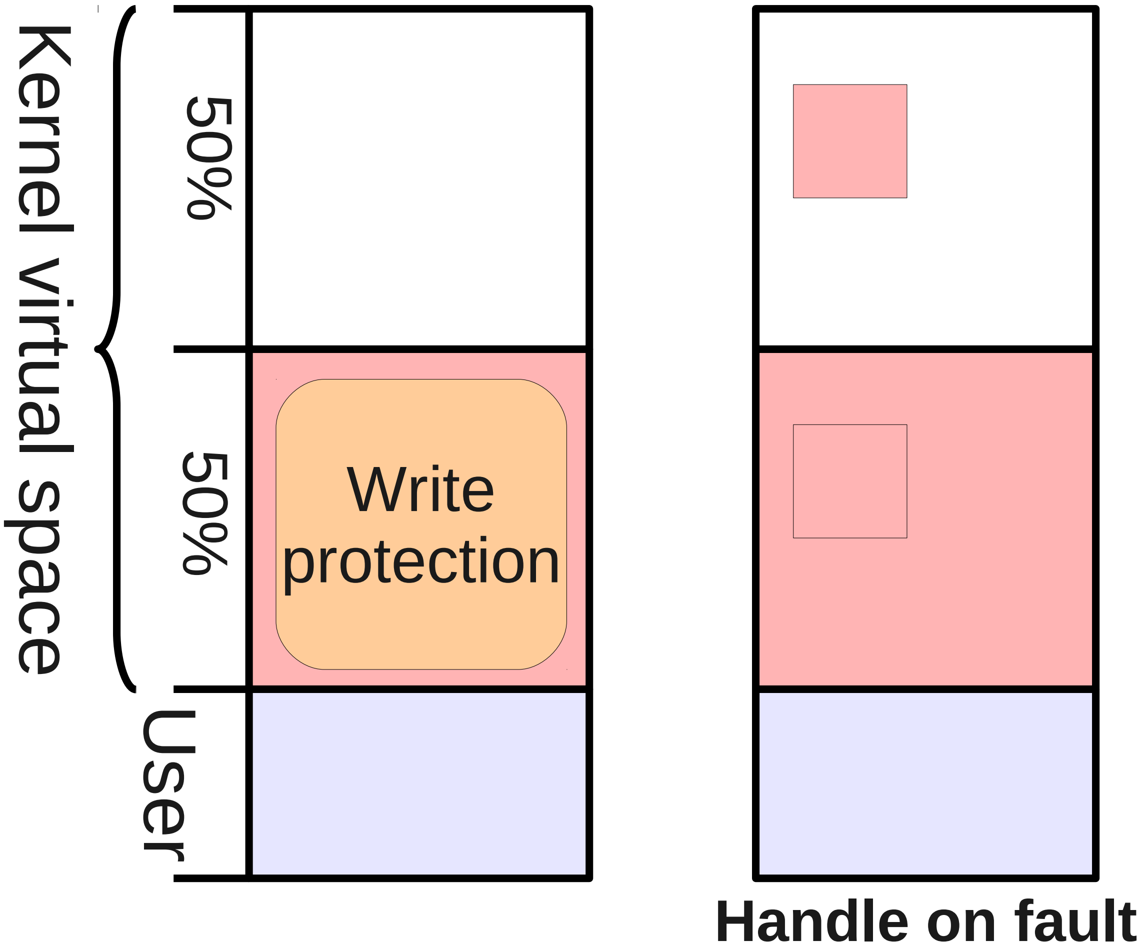


Handle on fault

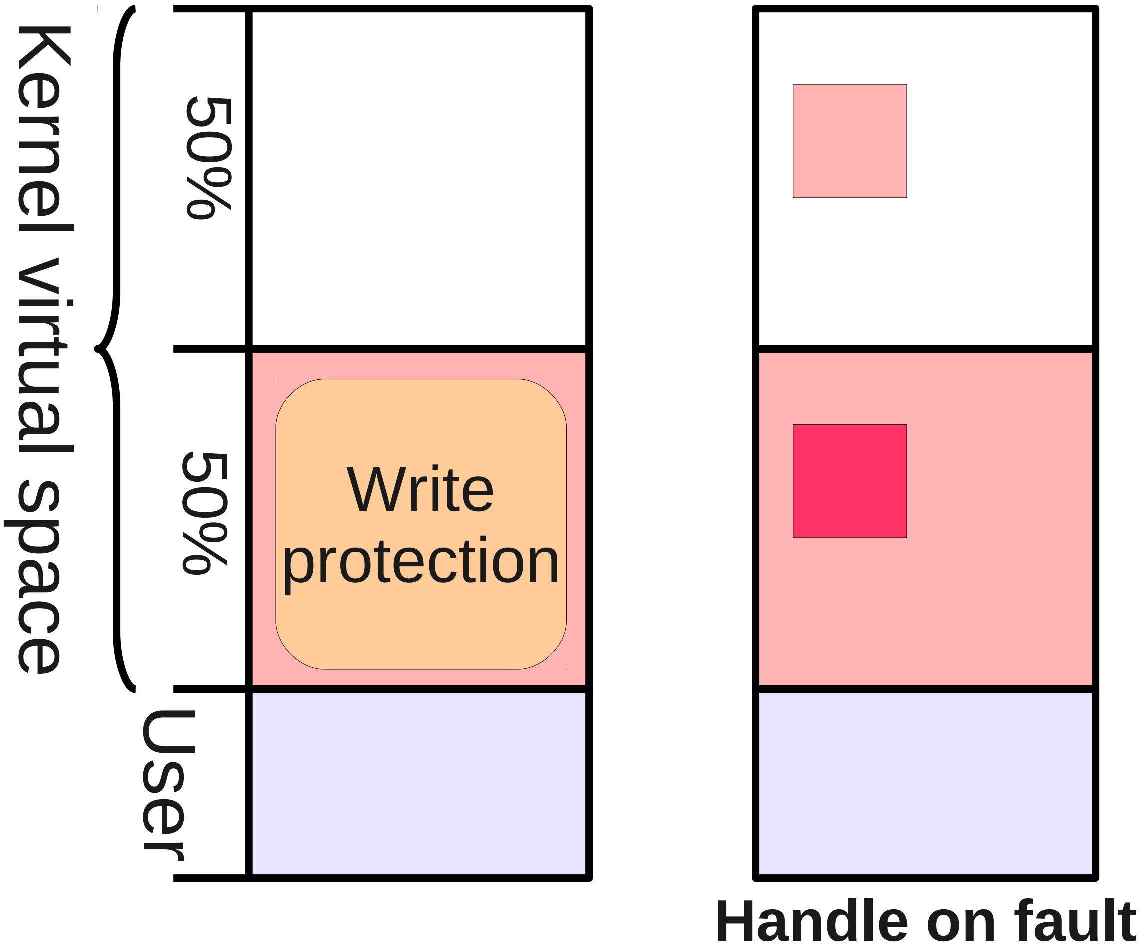
Flow of processing (2)



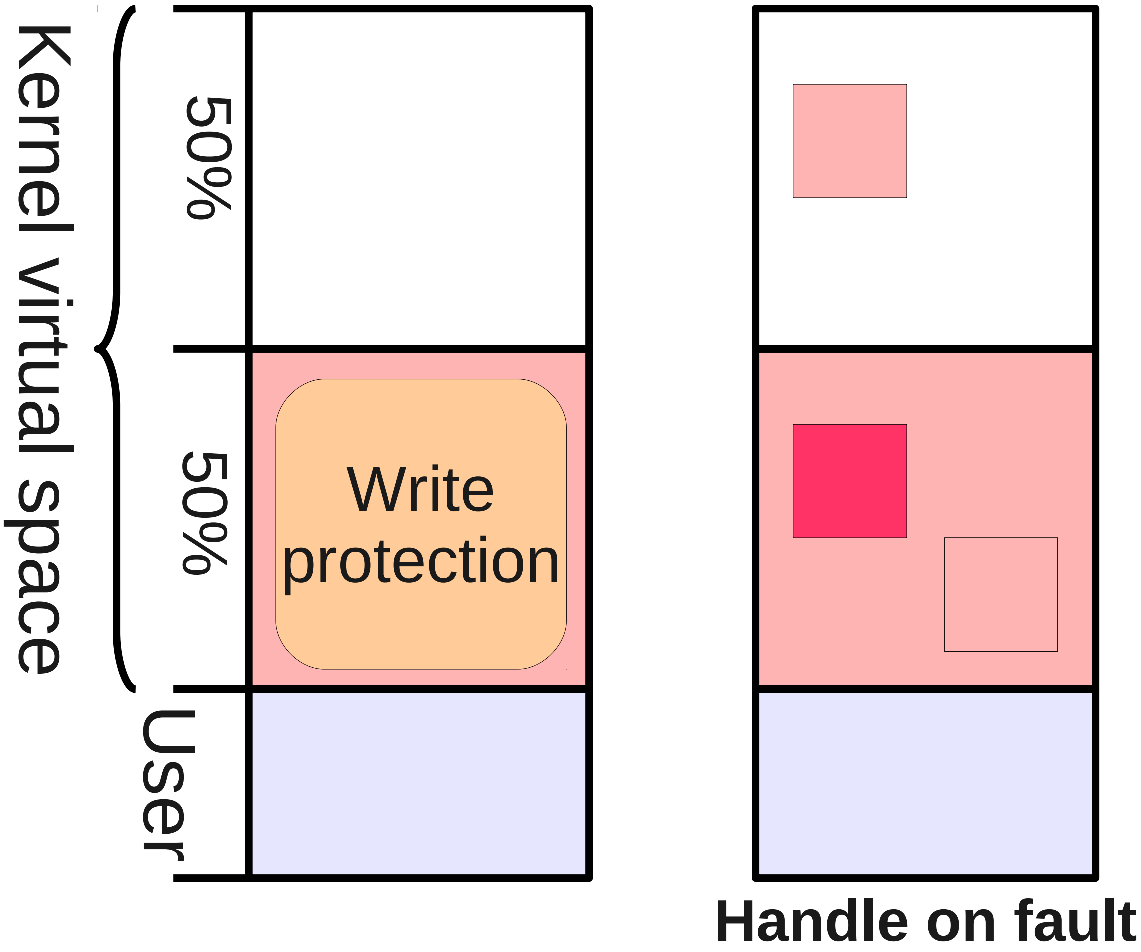
Flow of processing (2)



Flow of processing (2)

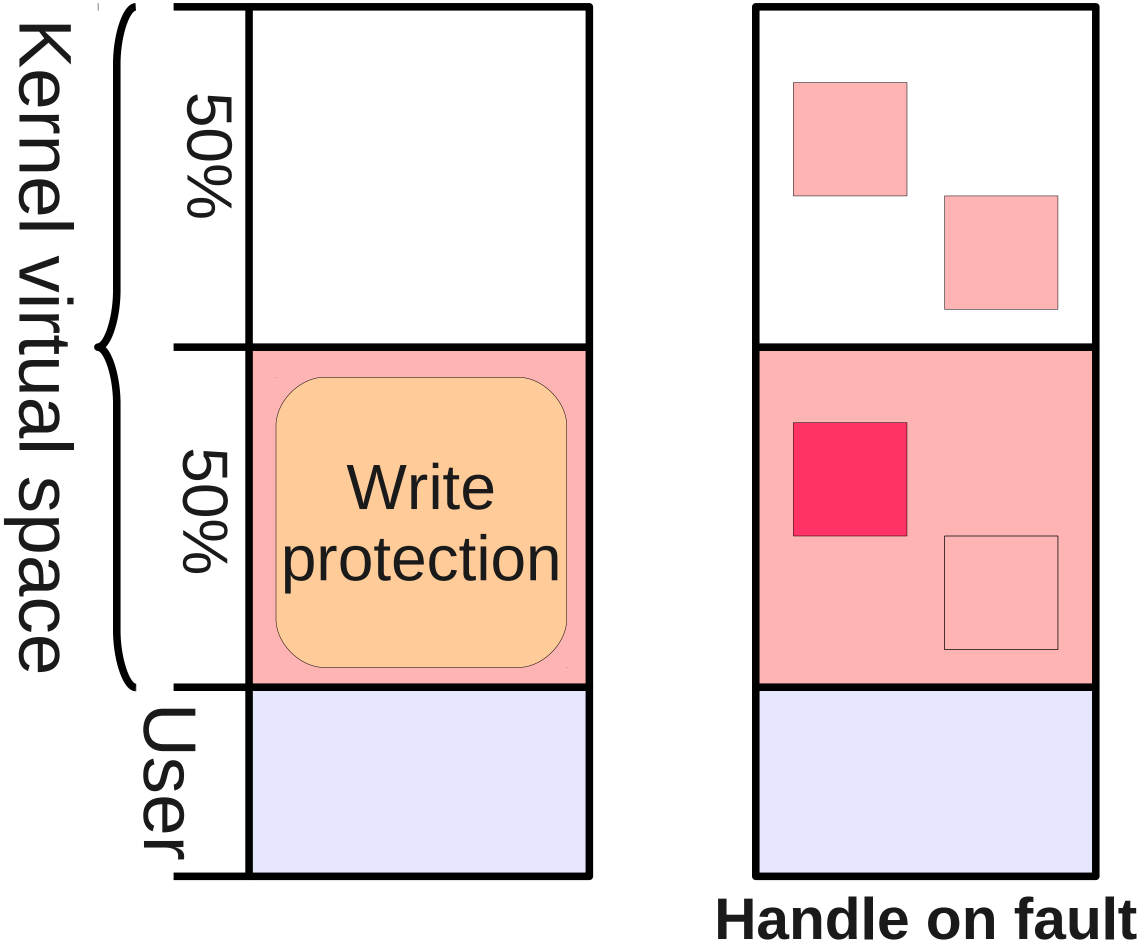


Flow of processing (2)

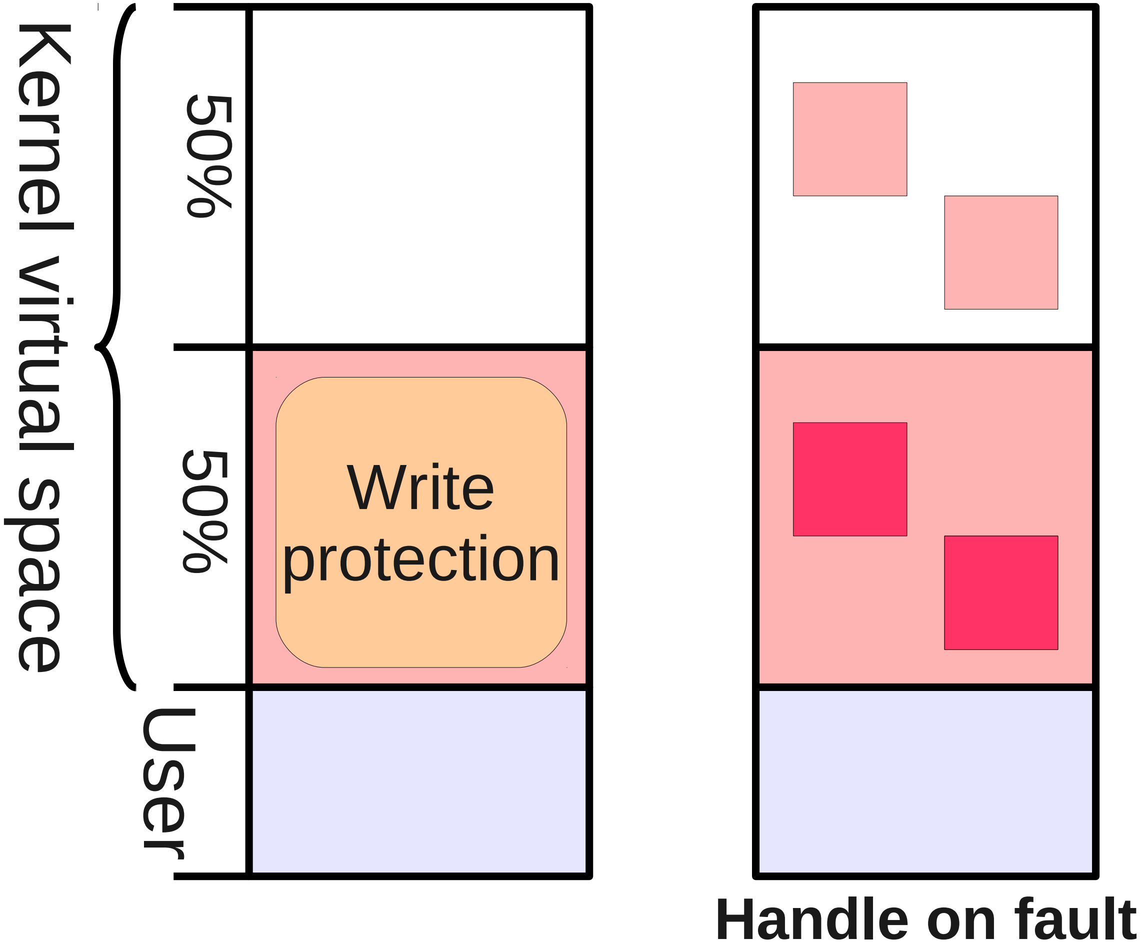


Copyright (c) 2012 Hitachi LTD., Yokohama Research Lab. All rights reserved.

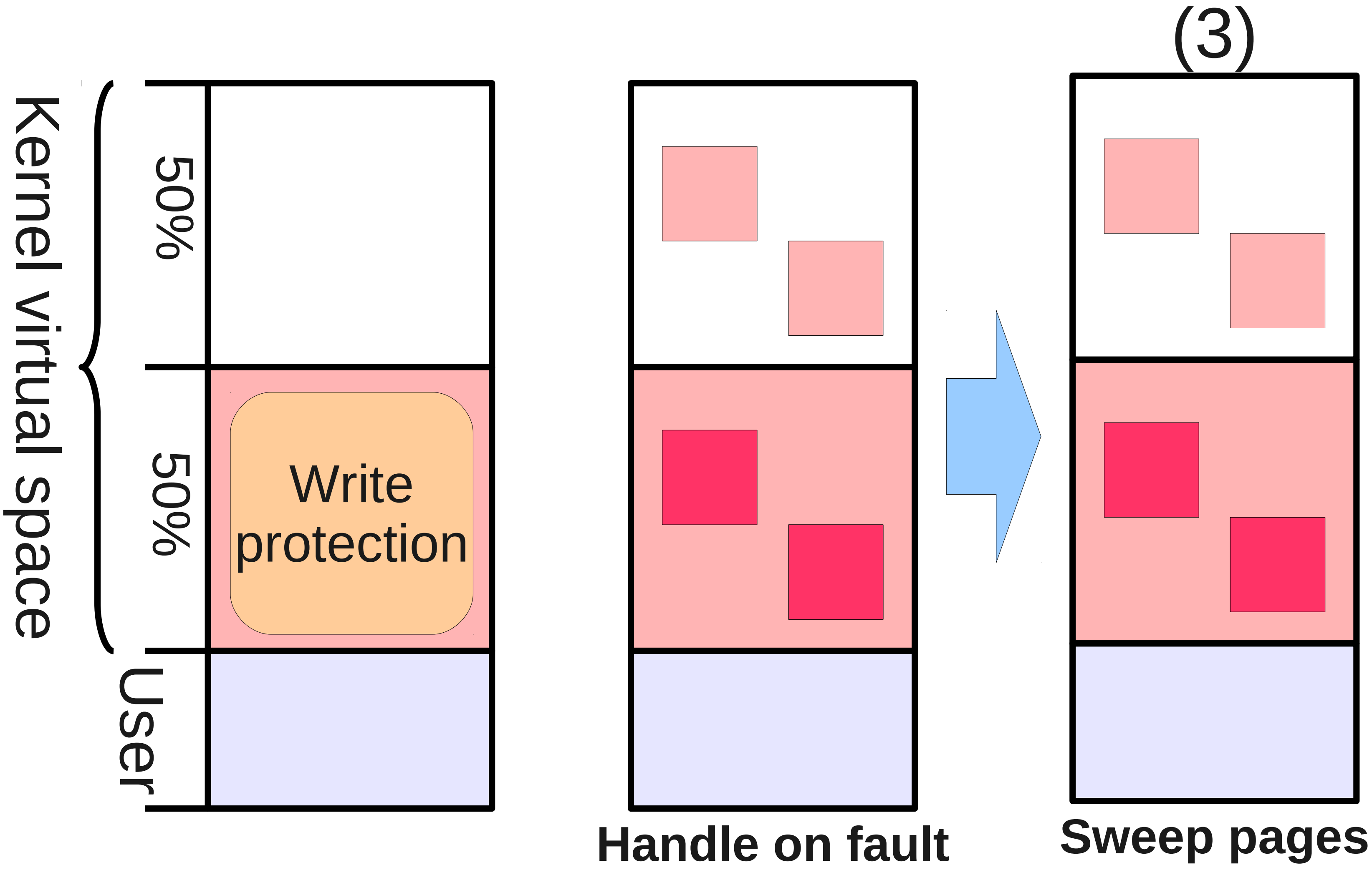
Flow of processing (2)



Flow of processing (2)

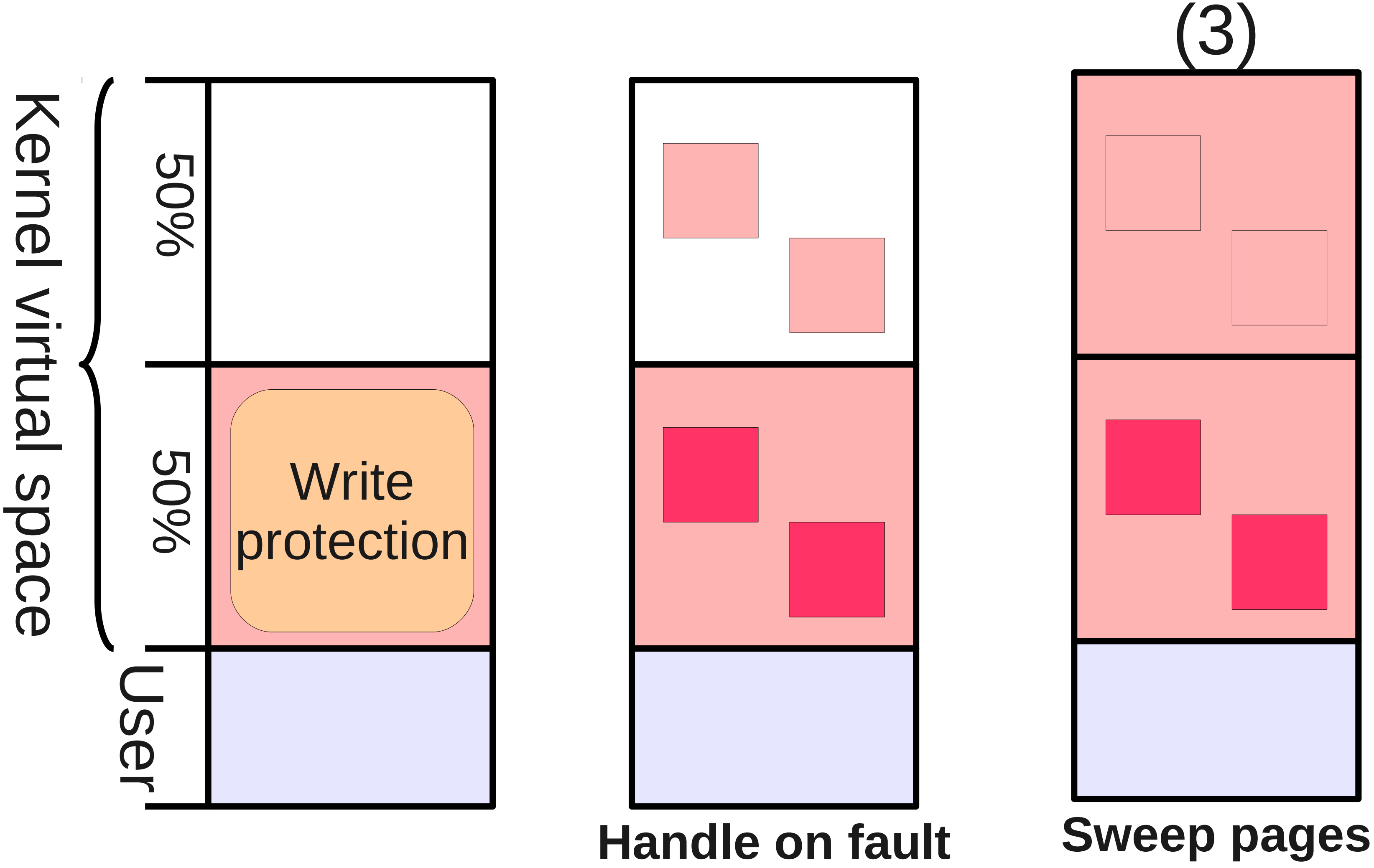


Flow of processing



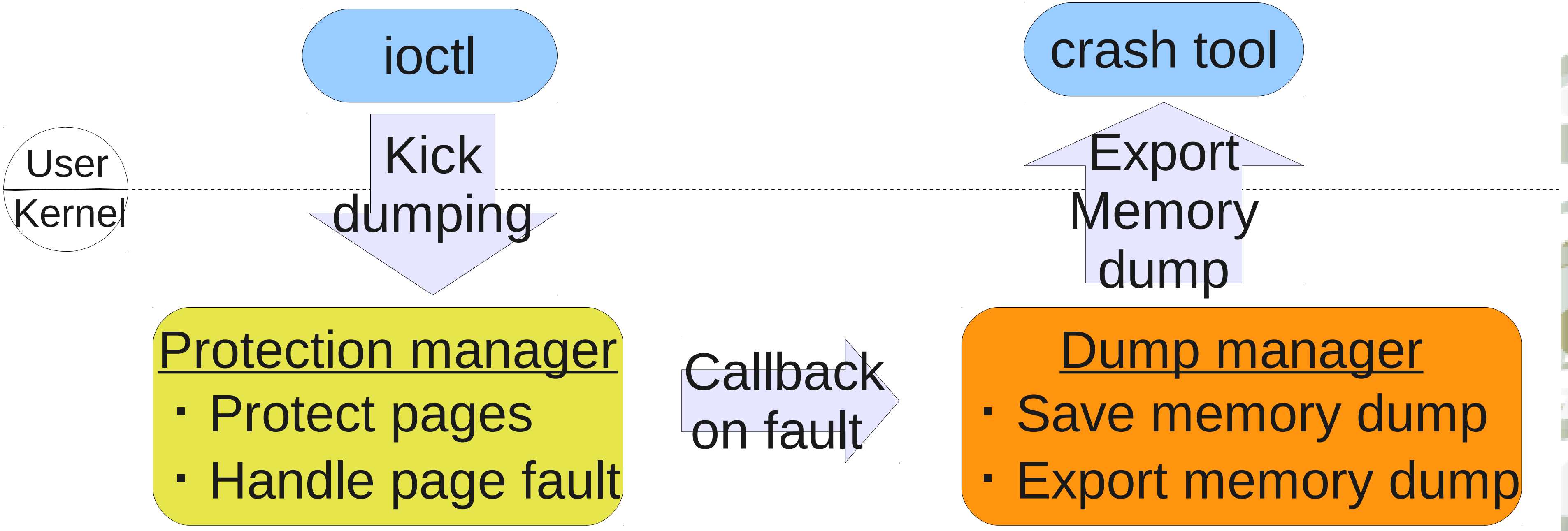
Copyright (c) 2012 Hitachi LTD., Yokohama Research Lab. All rights reserved.

Flow of processing



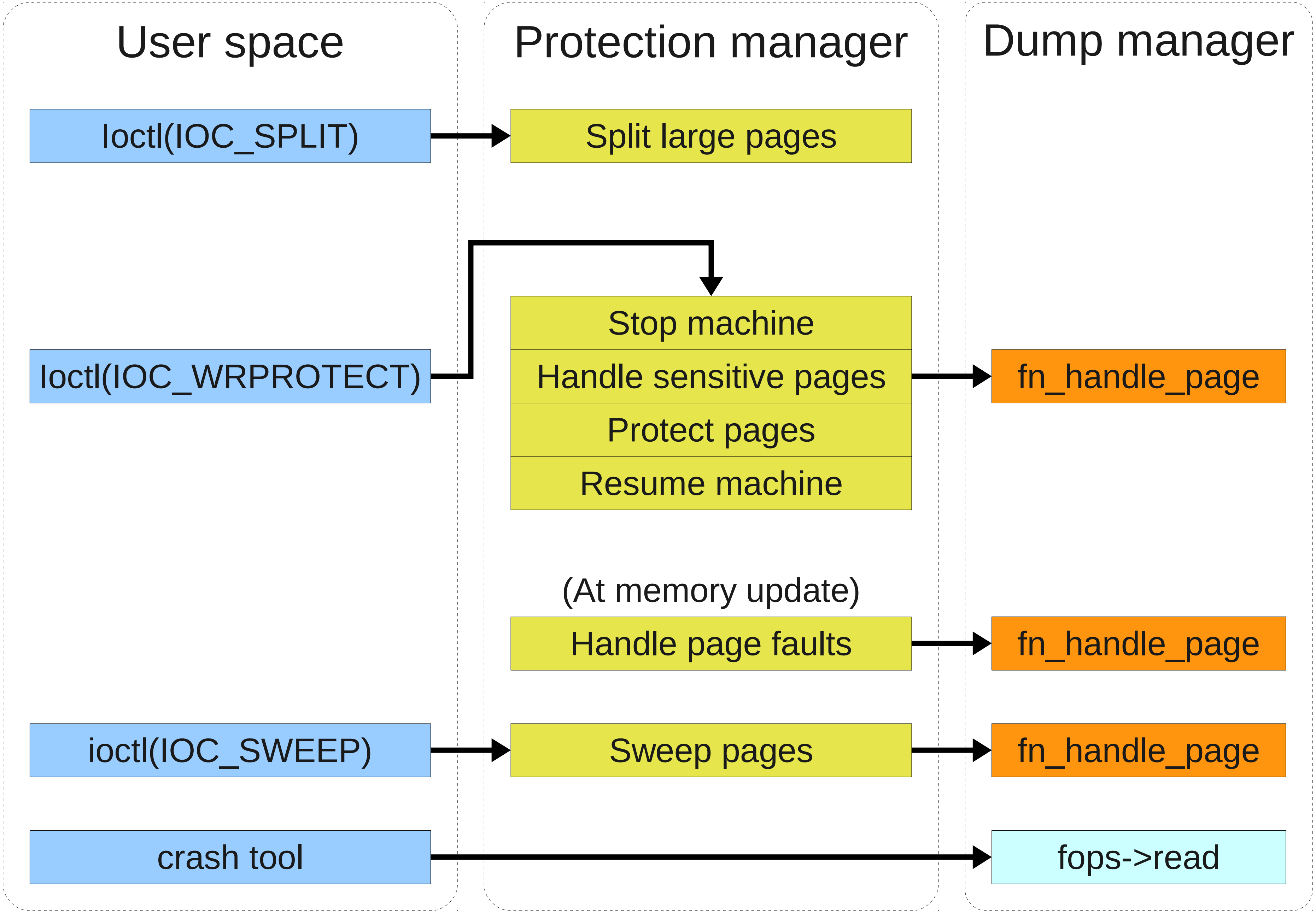
Overview of “livedump”

- 2 parts
  - Protection manager
  - Dump manager



Status of development

- 1150-line prototype with many limitations
- Limitation of Protection manager
    - Only supports protection of kernel space
    - Only supports x86-64 architecture
    - Only supports 4K pages
        - Need to split large pages into 4K ones in advance.
- Limitation of Dump manager
    - Need to allocate 50% of RAM to store dump.



User space

Protection manager

Dump manager

Ioctl(IOC_SPLIT)

Split large pages

Ioctl(IOC_WRPROTECT)

Stop machine

Handle sensitive pages

Protect pages

Resume machine

fn_handle_page

(At memory update)

Handle page faults

fn_handle_page

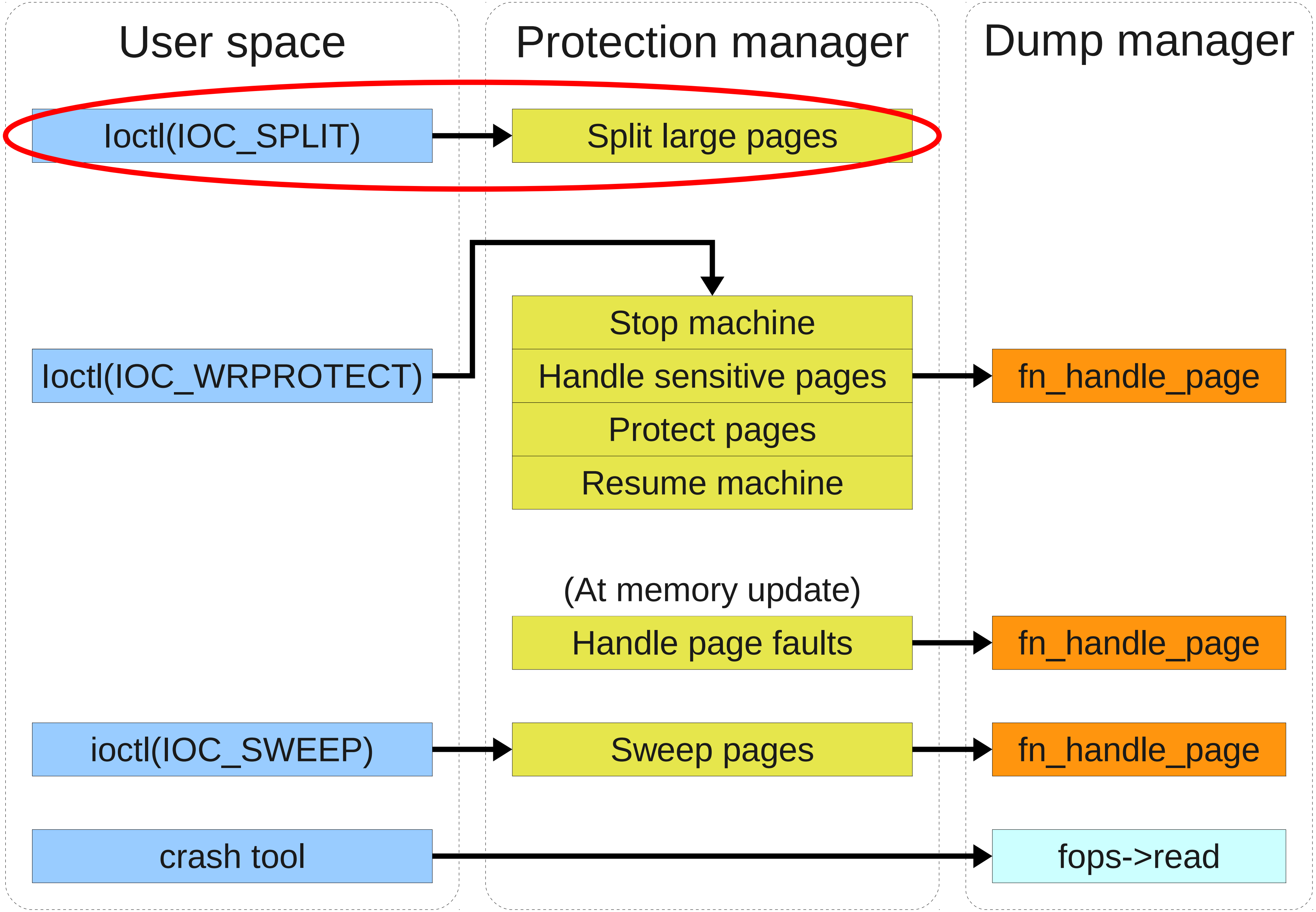
ioctl(IOC_SWEEP)

Sweep pages

fn_handle_page

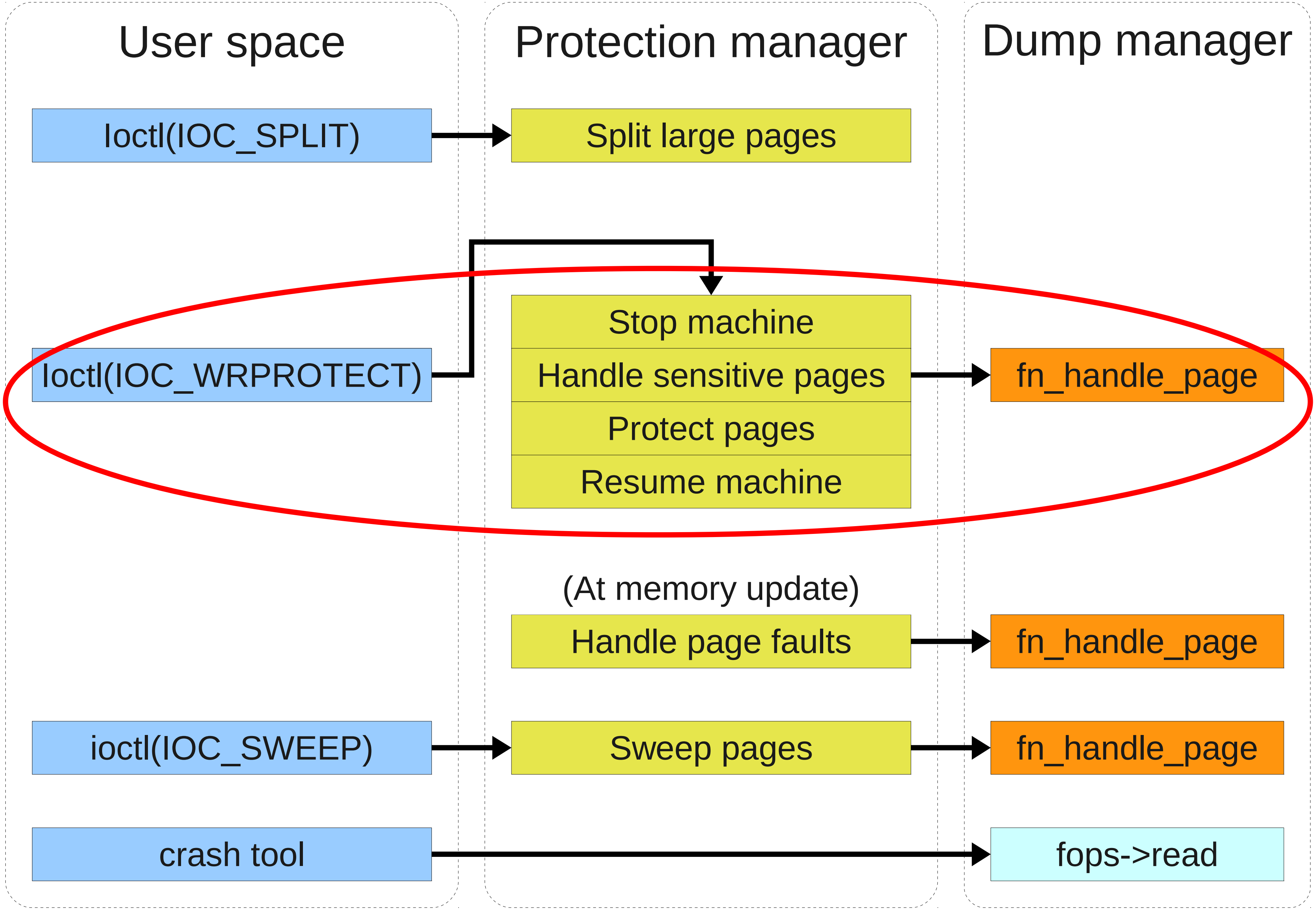
crash tool

fops->read


User space
Protection manager
Dump manager
Ioctl(IOC_SPLIT)
Split large pages
Ioctl(IOC_WRPROTECT)
Stop machine
Handle sensitive pages
Protect pages
Resume machine
fn_handle_page
(At memory update)
Handle page faults
fn_handle_page
ioctl(IOC_SWEEP)
Sweep pages
fn_handle_page
crash tool
fops->read

Split large pages

- Livedump can only protect 4K pages.
  - Splitting a large (2M or 1G) page during page fault handling is under development.
- At the moment, all large pages need to be split into 4K pages by set_memory_4k() in advance.
- This step will be unnecessary in the future.



User space

Protection manager

Dump manager

Ioctl(IOC_SPLIT)

Split large pages

Ioctl(IOC_WRPROTECT)

Stop machine

Handle sensitive pages

Protect pages

Resume machine

fn_handle_page

(At memory update)

Handle page faults

fn_handle_page

ioctl(IOC_SWEEP)

Sweep pages

fn_handle_page

crash tool

fops->read

Write protection

- Sequence
  1. Stop machine
  2. Handle sensitive pages
  3. Protect pages
  4. Resume machine

Stop machine

- We need a consistent memory snapshot, and so must protect all pages while processing is suspended.
- Livedump simply uses stop_machine() for this purpose.
- SMP in stop machine
  - Leader CPU: Protect pages
  - All CPUs: Wait for leader's job, and then flush TLBs

Handle sensitive pages

- Sensitive pages
  - Following pages cannot be protected.
    - Pages that can be updated in PF handling (This leads to Infinite loop of PF)
    - Pages that can be updated in NMI handling (This leads to nested NMI handling => panic)
  - Such pages are:
    - Kernel/Exception/Interrupt stacks
    - Page table structure
    - task_struct
    - .data section of kernel
    - per_cpu areas

Handle sensitive pages (Cont'd)

- Livedump dumps sensitive pages in stop machine.
- Via callback
  - `int fn_handle_page(unsigned long pfn);`
- `fn_handle_page(pfn)`
  - Defined by Dump manager.
  - Saves content of page.

Protect pages

- Manipulation of PTEs(Page Table Entries)

1. Copy _PAGE_RW flag to _PAGE_ORG_RW flag
   - `#define _PAGE_ORG_RW _PAGE_UNUSED1`
   - Both flags are in each PTE.
   - This is needed because there can be originally read-only pages.
2. Clear _PAGE_RW flag
   - Make a page read-only.

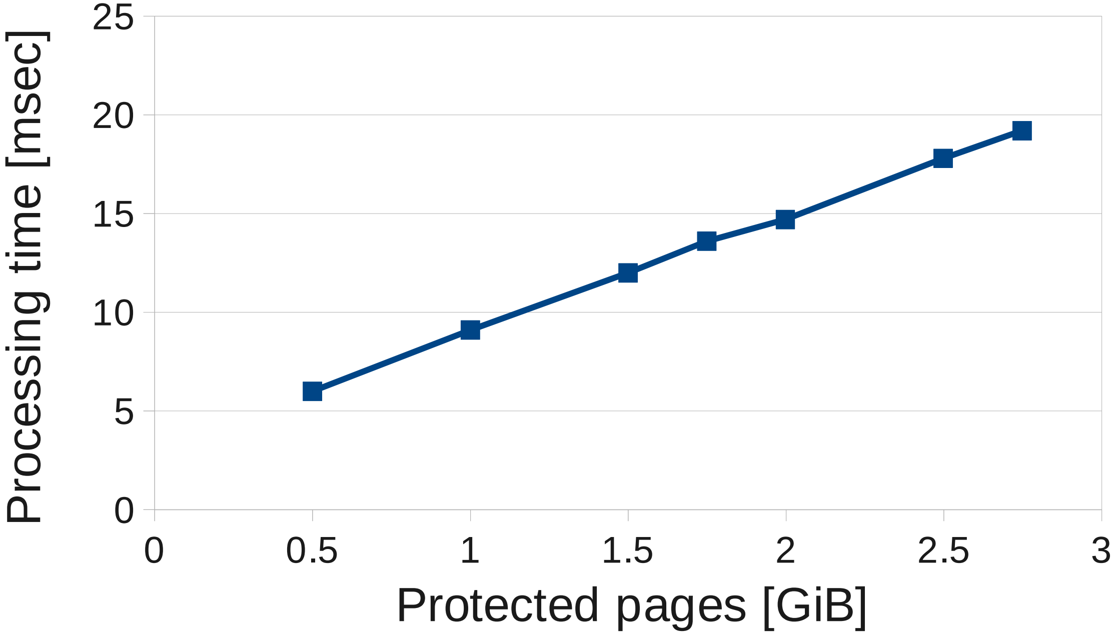
How long does it take?

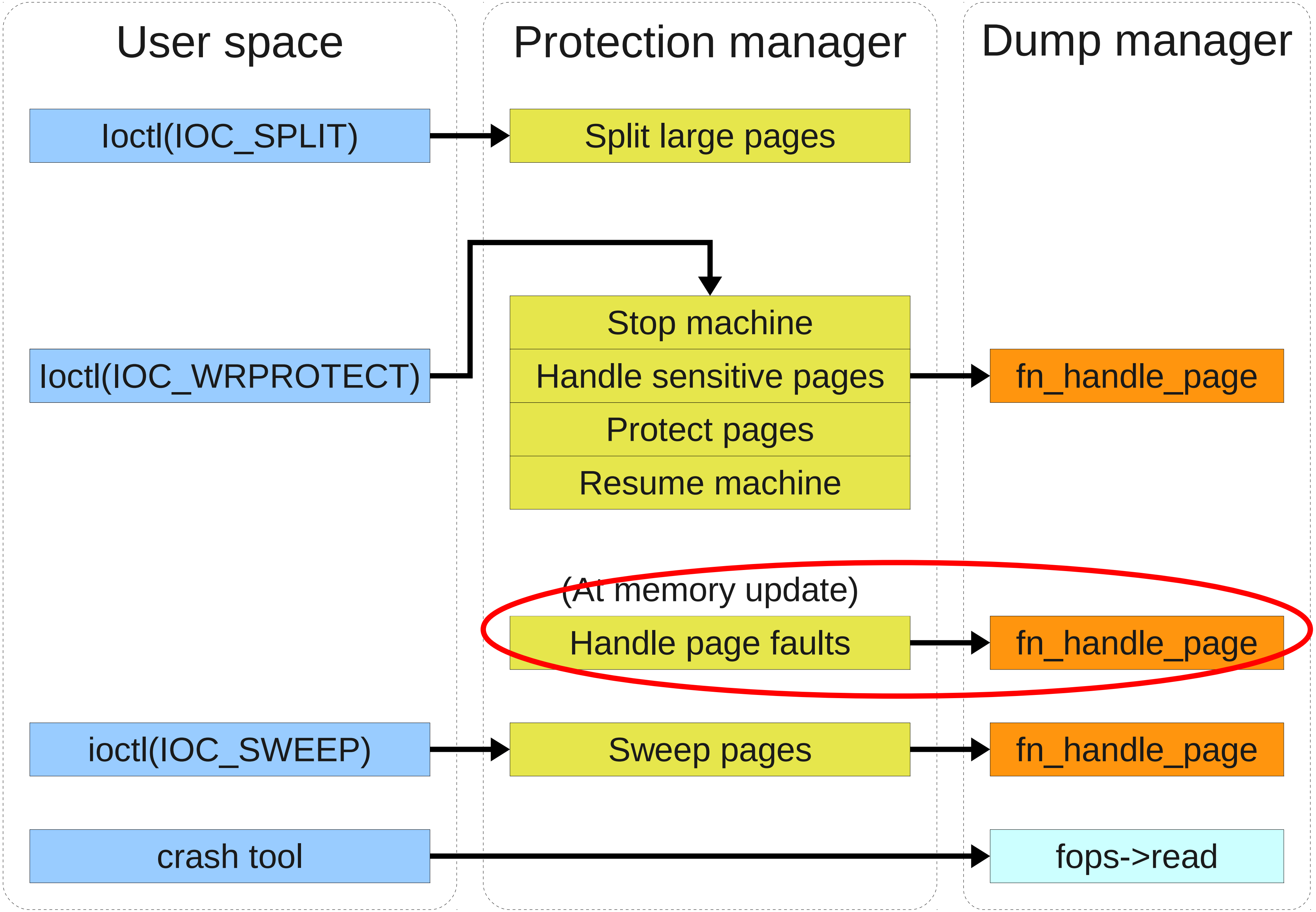
- Processing of write protection
  1. Stop machine
  2. Handle sensitive pages
  3. Protect pages
  4. Resume machine

Down time
(Stop machine)

How long does it take? (Cont'd)

- Down time
  - Is measured with Xeon W3520 @ 2.67 Ghz
  - Increases by 6 msec/GiB





User space

Protection manager

Dump manager

Ioctl(IOC_SPLIT)

Split large pages

Ioctl(IOC_WRPROTECT)

Stop machine

Handle sensitive pages

Protect pages

Resume machine

fn_handle_page

(At memory update)

Handle page faults

fn_handle_page

ioctl(IOC_SWEEP)

Sweep pages

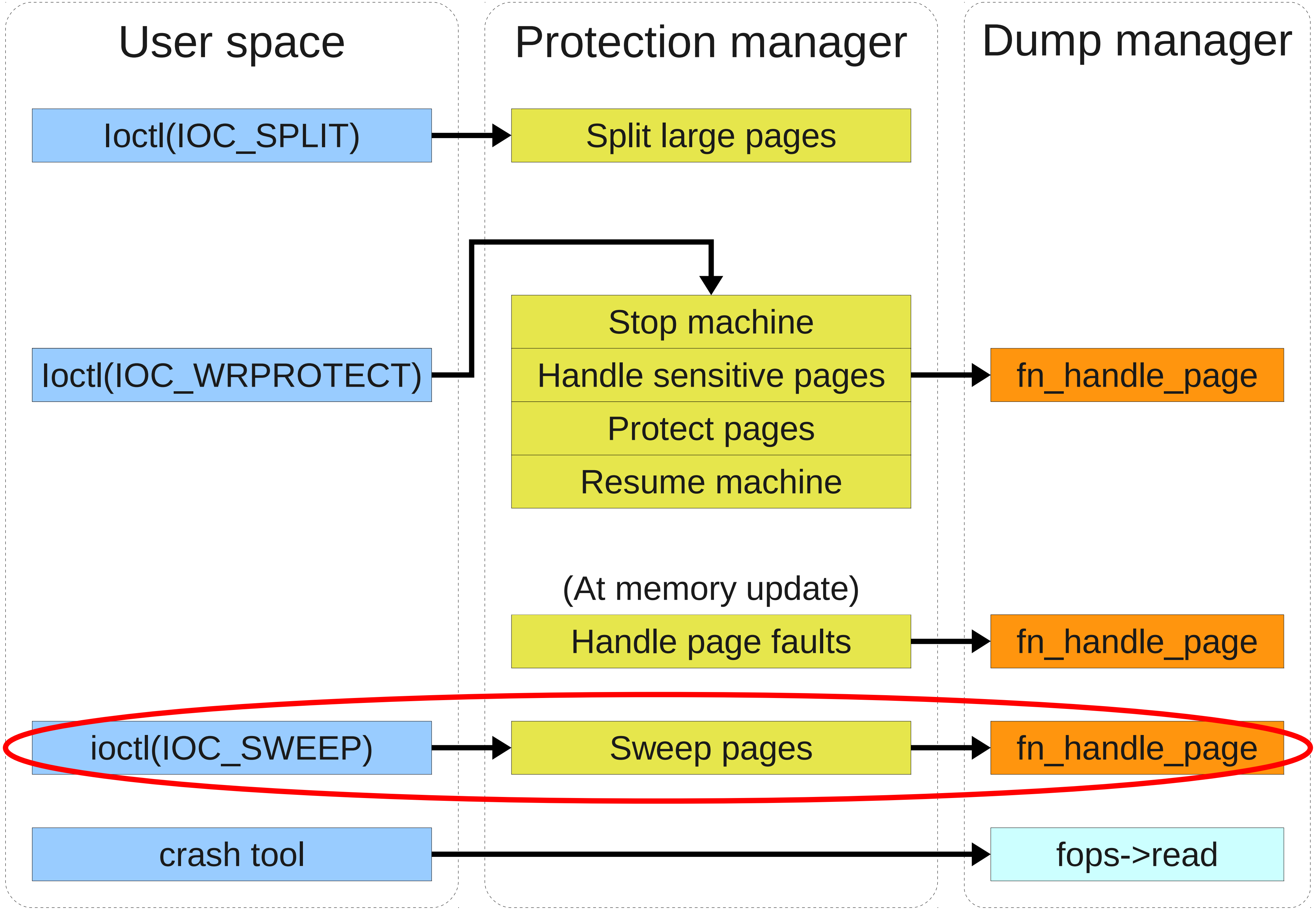
fn_handle_page

crash tool

fops->read

Handle page faults

- Livedump's notifier-call-chain in do_page_fault
  - Check cause of fault
    - Test _PAGE_ORG_RW flag
  - Exclusion control
    - test_and_clear_bit(pfn, pgbmp)
  - Dump the page
    - fn_handle_page(pfn)
  - Unprotect the page
    - Copy back _PAGE_ORG_RW flag to _PAGE_RW flag
    - Clear _PAGE_ORG_RW flag



User space

Protection manager

Dump manager

Ioctl(IOC_SPLIT)

Split large pages

Ioctl(IOC_WRPROTECT)

Stop machine

Handle sensitive pages

Protect pages

Resume machine

fn_handle_page

(At memory update)

Handle page faults

fn_handle_page

ioctl(IOC_SWEEP)

Sweep pages

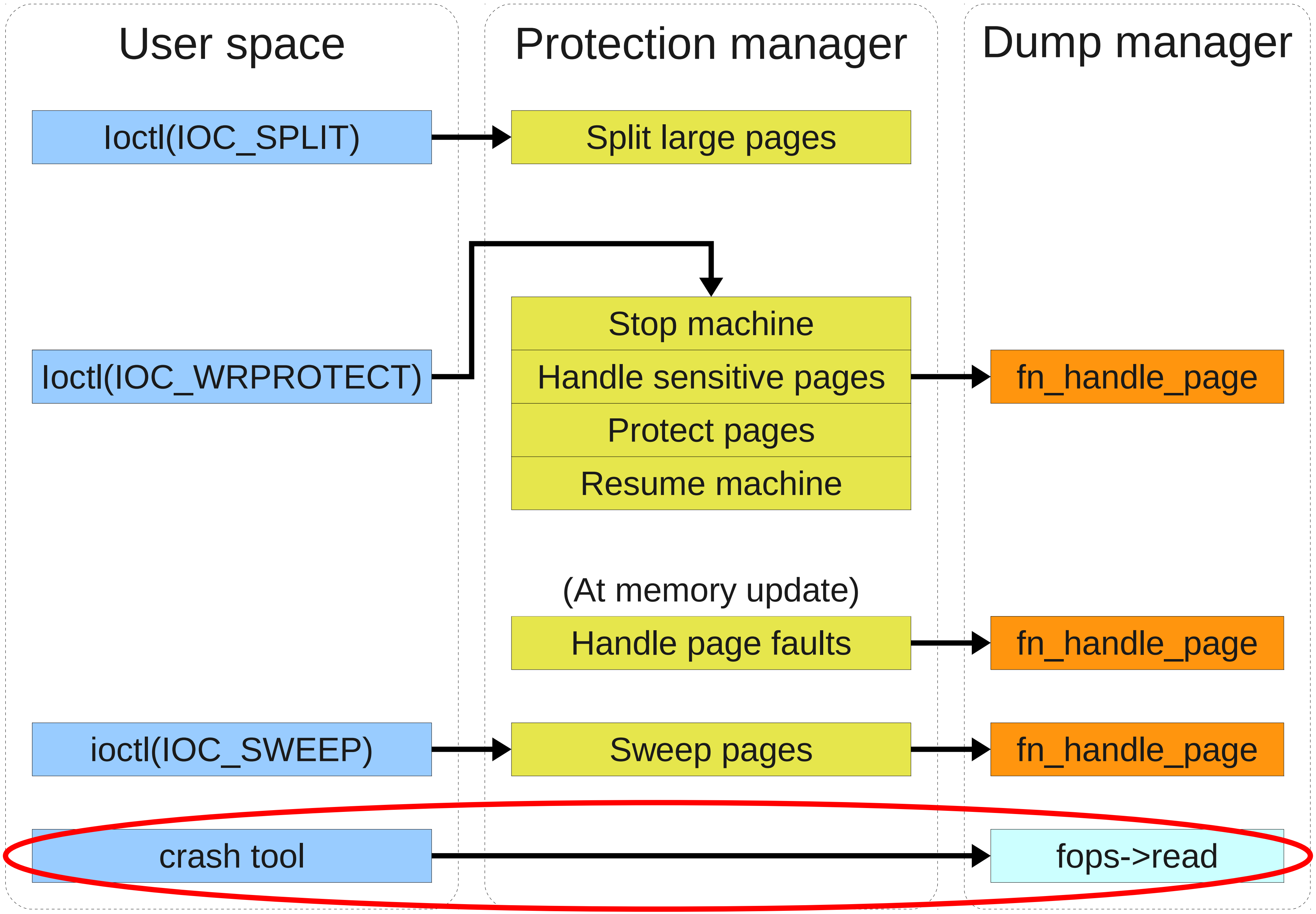
fn_handle_page

crash tool

fops->read

Sweep pages

- Batch dumping of all pages not dumped yet.
  - via `fn_handle_page()` in turn



User space

Protection manager

Dump manager

Ioctl(IOC_SPLIT)

Split large pages

Ioctl(IOC_WRPROTECT)

Stop machine

Handle sensitive pages

Protect pages

Resume machine

fn_handle_page

(At memory update)

Handle page faults

fn_handle_page

ioctl(IOC_SWEEP)

Sweep pages

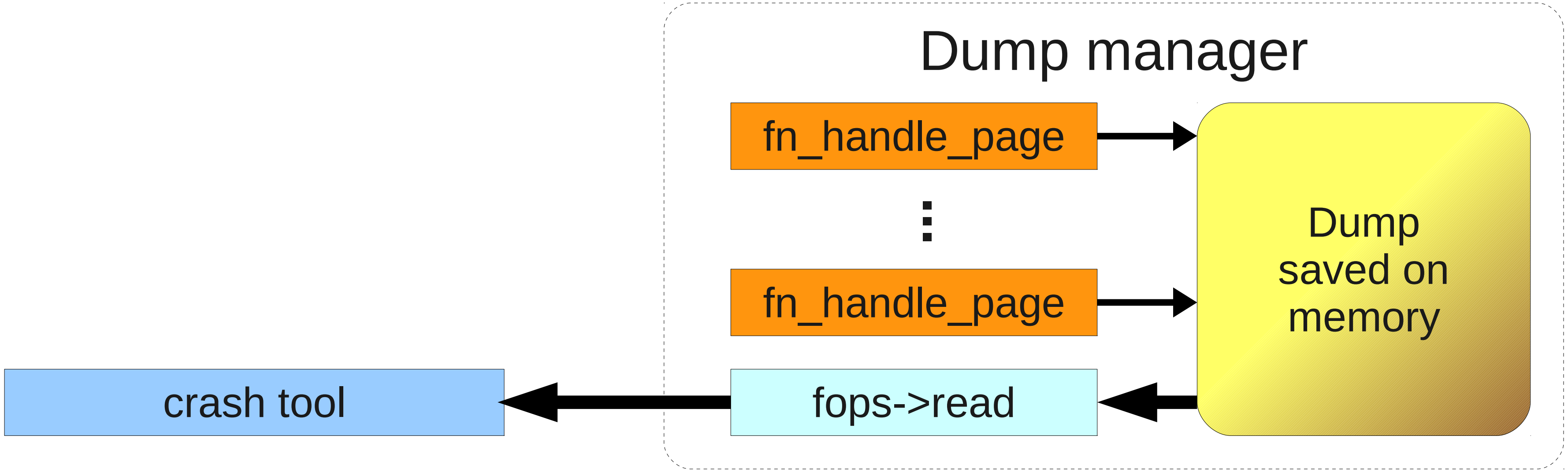
fn_handle_page

crash tool

fops->read

Dump analysis via crash

- Dumped pages are exported as a character device.
  - `fops->read()`
  - `fops->llseek()`
- You can use crash with the character device.



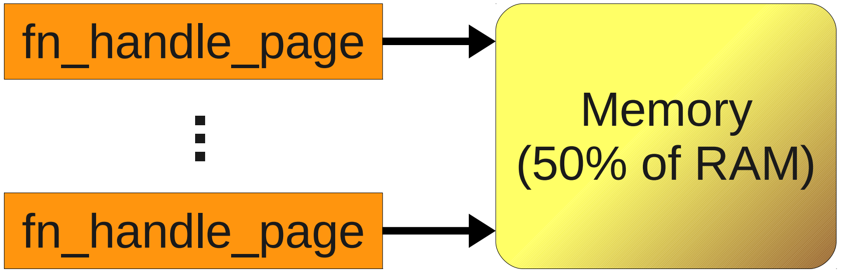
1. What is Live Dump?
2. Implementation
3. Future work

Large page support

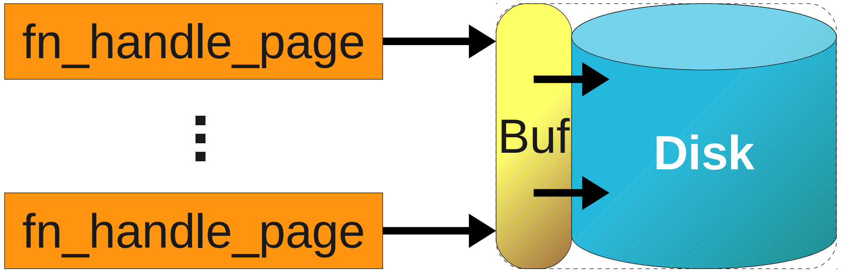
- Support write-protection of 2M pages
    - To reduce TLB consumption
- Fix splitting phase
    - Only 1G pages are split to 2M x 512 pages.
    - 2M pages remain 2MB.
- Fix page fault handling
    - 2MB data are copied on page fault.
        - Copy cost = 200usec (on my Xeon machine)

On-the-fly dumping to disk

- At the moment



- On-the-fly version



Conclusion

- I developed the prototype of Live Dump
  - Memory dump with a running OS
  - Technique based on CoW
- Performance (down time)
  - 15 ms with 2GB and increases by 6 ms / GB.
- Limitation
  - It has many many limitations...

RFC patchset

- RFC patchset of livedump has been submitted.
  - On May 25

- Please give me feedback!

Trademarks

- 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.

Thank you!