Detecting silent data corruptions and memory leaks using DMA Debug API

Shuah Khan
Senior Linux Kernel Developer – Open Source Group
Samsung Research America (Silicon Valley)
shuah.kh@samsung.com

Abstract

Linux kernel drivers map and unmap Dynamic DMA buffers using DMA API. DMA map operations can fail. Failure to check for errors can result in a variety of problems ranging from panics to silent data corruptions. Kernel panics can be fixed easily, however data corruptions are hard to debug.

DMA mapping error analysis performed by the presenter found that more than 50% of map interface return values go unchecked in the kernel. Further more, several drivers fail to unmap buffers when an error occurs in the middle of a multi-page dma mapping attempt.

Presenter added a new DMA Debug interface in Linux 3.9 to check for missing mapping error checks.

This talk will share the results of the analysis and discuss how to find and fix missing mapping errors checks using the new interface. This talk will discuss possible enhancements to DMA Debug API to detect and flag unmap errors.

Agenda

- DMA API and its usage rules
- DMA-debug API
- DMA-debug API – what is missing?
- Why check dma mapping errors?
- Analysis results
- After debug_dma_mapping_error()
- Checking mapping errors (examples: incorrect and correct)
- Use or not use unlikely()
- dma_mapping_error()
- Why unmap after use?
- Next steps – possible enhancements to DMA-debug API
- Questions

DMA API

- Linux kernel supports Dynamic DMA mapping
  - dma_map_single() and dma_unmap_single()
  - dma_map_page() and dma_unmap_page()
  - dma_map_sg() and dma_unmap_sg()
- References:
  - https://www.kernel.org/doc/Documentation/DMA-API.txt
  - https://www.kernel.org/doc/Documentation/DMA-API-HOWTO.txt
  - https://www.kernel.org/doc/Documentation/DMA-attributes.txt

DMA API usage rules

- Drivers
  - can map and unmap DMA buffers at run-time
  - should map buffers, use, and unmap when buffers are no longer needed – don't hoard buffers
  - ensure all mapped buffers are unmapped
  - should use generic DMA API as opposed bus specific DMA API e.g: pci_dma_*()
  - should call dma_mapping_error() to check for mapping errors before using the returned dma handle

DMA-debug API

- designed for debugging driver DMA API usage errors
- keeps track of DMA mappings per device
- debug_dma_map_page() - adds newly mapped entry to keep track. Sets flag to track missing mapping error checks
- detects missing mapping error checks in driver code after DMA mapping.
- debug_dma_mapping_error() - checks and clears flag set by debug_dma_map_page()

DMA-debug API

- detects unmap attempts on invalid dma addresses
- generates warning message for missing dma_mapping_error() calls with call trace leading up to dma_unmap()
- debug_dma_unmap_page() - checks if buffer is valid and checks dma mapping error flag
- CONFIG_HAVE_DMA_API_DEBUG and CONFIG_DMA_API_DEBUG enabled

DMA-debug API

CONFIG_DMA_API_DEBUG disabled

dma_mapping_error()

debug_dma_mapping_error() is a stub

CONFIG_DMA_API_DEBUG enabled

dma_mapping_error()

debug_dma_mapping_error()
code is executed to clear mapping error
flag set by debug_dma_map()

DMA-debug API – what is missing?

- Missing: Detecting missing unmap cases that would result in dangling DMA buffers
- Weakness: Detecting missing mapping errors is done in debug_dma_unmap_page().
  - These go undetected when driver fails to unmap.

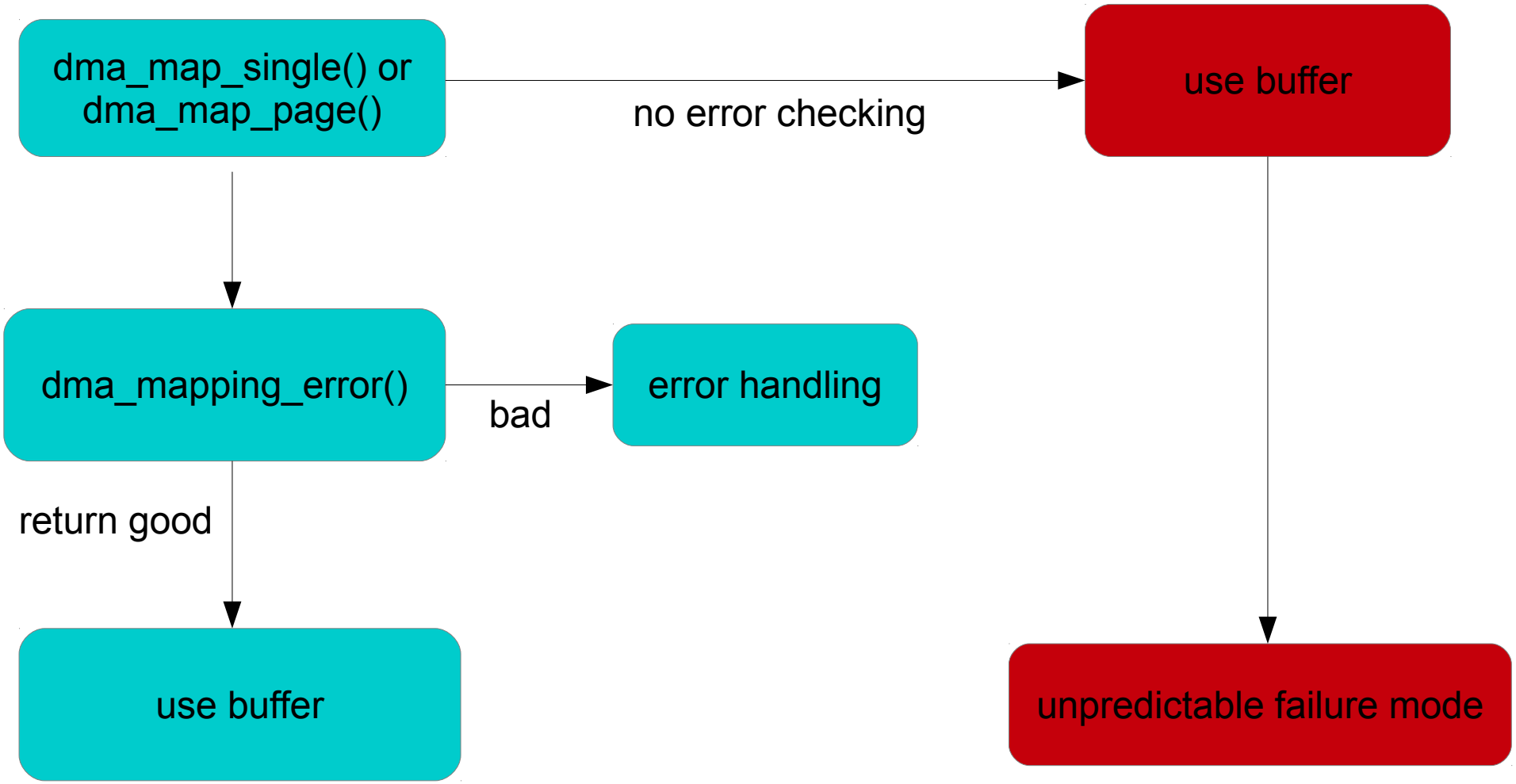
Why check dma mapping errors?

- Failure to check mapping error prior to using the address
  - could result in panics, silent data corruptions
  - panics could be fixed easily once they occur
  - data corruptions are very hard to debug, not to mention the damage they do.

Why check dma mapping errors?

- Preventing errors is better than the alternative
- Detection allows
  - Taking corrective action that is right for the condition
  - Prevents uncertain failure modes

DMA API error handling



Analysis results

- First analysis (linux-next August 6 2012):
  - large % (> 50%) of addresses returned by dma_map_single() and dma_map_page() go unchecked
- Current status (linux 3.12-rc5) October 2013:
  - large % (> 50%) of addresses returned by dma_map_single() and dma_map_page() go unchecked
- No change in the % of missing mapping error checks.

Problem classification - broken

- Broken
  - no dma mapping error checks done on the returned address.
- Partially Broken
  - not all dma_map_single() and dma_map_page() calls are followed by mapping error checks.
- Unmap broken
  - checks dma mapping errors
  - doesn't unmap already mapped pages when mapping error occurs in the middle of a multiple page mapping attempt.

Problem classification - good

- Good
  - checks dma mapping errors correctly
  - checks dma mapping errors with unlikely()
  - unmaps already mapped pages when mapping error occurs in the middle of a multiple page mapping attempt.

dma_map_single() results

- Broken - 46%
- Partially broken - 11%
- Unmap broken - 6%
- Good - 35%

dma_map_page() results

- Broken - 59%
- Partially broken - 11%
- Unmap broken - 15%
- Good - 19%

Things considered and action taken

- When do mapping errors get detected?
- How often do these errors occur?
- Why don't we see failures related to missing dma mapping error checks?
- Are they silent failures?
- What is done - a new DMA-debug interface is added after the first analysis

After debug_dma_mapping_error()

- debug_dma_mapping_error() went into Linux-3.9
- Several drivers have been fixed as a result of the warnings.
- Intel drivers deserve a special mention.
  - drivers flagged in the first analysis have been fixed.
- New code and drivers are added that fail to check errors since the last analysis

Checking mapping errors

Incorrect example 1:

- Non-generic and not portable - depends on architecture specific dma implementations

```
dma_addr_t dma_handle;
dma_handle = dma_map_single(dev, addr, size, direction);
if ((dma_handle & 0xffff != 0) || (dma_handle >= 0x1000000)) {
        goto map_error;
}
```

Checking mapping errors

Incorrect example 2:

- Non-generic and not portable - depends on architecture specific DMA_ERROR_CODE definitions

```
dma_addr_t dma_handle;
dma_handle = dma_map_single(dev, addr, size, direction);
if (dma_handle == DMA_ERROR_CODE) {
        goto map_error;
}
```

Checking mapping errors

Generic and portable:

```
dma_addr_t dma_handle;
dma_handle = dma_map_page(dev, page, offset, size, direction);
if (dma_mapping_error(dev, dma_handle)) {
        /*
         * reduce current DMA mapping usage,
         * delay and try again later or
         * reset driver.
         */
        goto map_error_handling;
}
```

Use or not use unlikely()

likely() and unlikely() aren't efficient in all cases.
Don't use it especially in driver code.

```
if (unlikely(dma_mapping_error(dev, dma_handle))) {
        ---
}
```

More on this topic:

- https://lkml.org/lkml/2012/10/18/150
- http://blog.man7.org/2012/10/how-much-do-builtinexpect-likely-and.html

dma_mapping_error()

- It is implemented by all DMA implementations: the ones that don't implement, simply return 0
  - e.g: arch/openrisc/include/asm/dma-mapping.h
- Some architectures return DMA_ERROR_CODE
  - e.g: arch/sparc/include/asm/dma-mapping.h
- Some implement it invoking underlying dma_ops
  - e.g: arch/x86/include/asm/dma-mapping.h
- Good practice to use dma_mapping_error() to check errors and let the underlying DMA layer handle the architecture specifics.

Why unmap after use?

- Timely unmap of DMA buffers ensures buffer availability in need
- Failure to unmap when mapping error occurs in the middle of a multi-page DMA map attempt is a problem
  - equivalent to a memory leak condition
  - leaves dangling DMA buffers that will never get unmapped and reclaimed.
- Note: failure to unmap is not a problem on some architectures
  - however from drivers calling dma_unmap() is a good practice.

Next steps

- possible enhancements to DMA-debug API
  - enhance to check for unmap errors.
  - I am working on the following ideas – stay tuned
    - when should the unmap error check get triggered?
      - one possible option is when device object is released.
    - dynamic DMA-debug API?

- http://linuxdriverproject.org/mediawiki/index.php/User_talk:Shuahkhan

Questions?

Thank you.

Shuah Khan
Senior Linux Kernel Developer – Open Source Group
Samsung Research America (Silicon Valley)
shuah.kh@samsung.com