FUJITSU

shaping tomorrow with you

The Forefront of the Development for NVDIMM on Linux Kernel (Linux Plumbers conf. ver.)

2021/Sep/23

Yasunori Goto (Fujitsu Limited.)
Ruan Shiyang (Nanjing Fujitsu Nanda Software Technology Co., Ltd.)

Agenda

FUJITSU

- Summary of current status of development for NVDIMM (Yasunori)
  - Basis of NVDIMM on Linux
  - Issues of Filesystem-DAX (Direct Access mode)
- Deep dive to solve the issues of Filesystem-DAX (Ruan)
  - Support reflink & dedupe for fsdax
  - Fix NVDIMM-based Reverse mapping
- Conclusion

Self introduction

FUJITSU

- Yasunori Goto
  - I have worked for Linux and related OSS since 2002
    - Development for Memory hotplug feature of Linux Kernel
    - Technical Support for troubles of Linux Kernel
    - etc.
  - Currently, leader of Fujitsu Linux Kernel Development team
  - In the last few years, I have mainly worked for NVDIMM
    - some enhancement for RAS of NVDIMM
      - For Fault location feature
      - For Fault prediction feature

    "The ideal and reality of NVDIMM RAS"
    https://www.slideshare.net/ygotokernel/the-ideal-and-reality-of-nvdimm-ras-newer-version

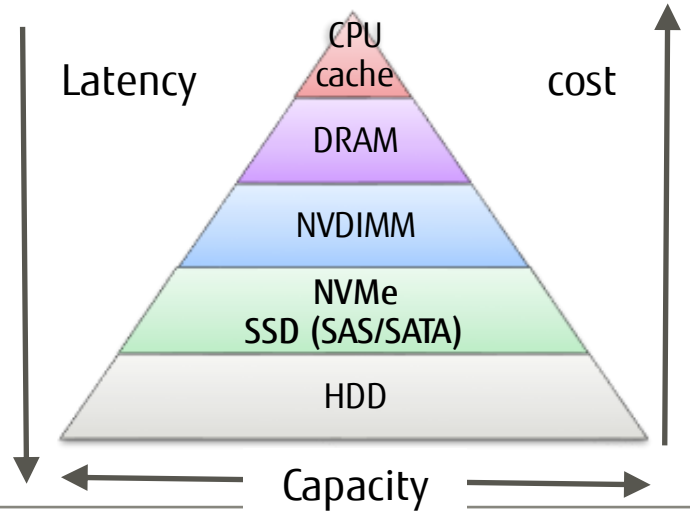
FUJITSU

Basis of NVDIMM on Linux

Characteristics of Non-Volatile DIMM (NVDIMM)

FUJITSU

- Persistent memory device which can be inserted to DIMM slot like DRAM
  - CPU can read/write NVDIMM directly
  - It can keep data persistency even if system is powered down or rebooted
  - Latency, capacity, and cost have characteristics between DRAM and NVMe
- Use case
  - Example
    - In memory Database
    - Hierarchical storage, distributed storage
    - Key-Value-store
- Famous Product
  - Intel Data Center Persistent Memory Module (DCPMM)



Impact of NVDIMM

FUJITSU

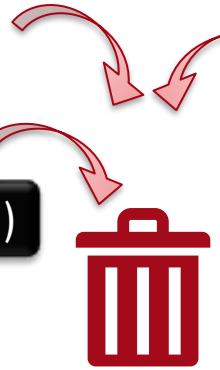
Traditional I/O layer becomes redundant for NVDIMM

Page cache

It was created for SLOW I/O storage, but NVDIMM is fast enough without page cache

Sync()/fsync()/msync()

If user process calls cpu flush instruction, then it is enough to make persistency



System call (Switch to kernel mode)

- Application can read/write to NVDIMM directly
- Even system call may be waste of time due to switching between kernel mode and user mode

New interface is expected for NVDIMM!

NVDIMM is difficult for traditional software

FUJITSU

- Many software assumes that memory is VOLATILE yet

What will be necessary for a software to use NVDIMM?

Need to prepare for sudden powerdown

- In older CPU, its cache is still volatile
- If system power down suddenly, then some data may not be stored

Need data structure compatibility on NVDIMM

- Should not change data structures in NVDIMM
- If the structures are changed, software update will be cause of disaster

Need to detect / correct collapsing data

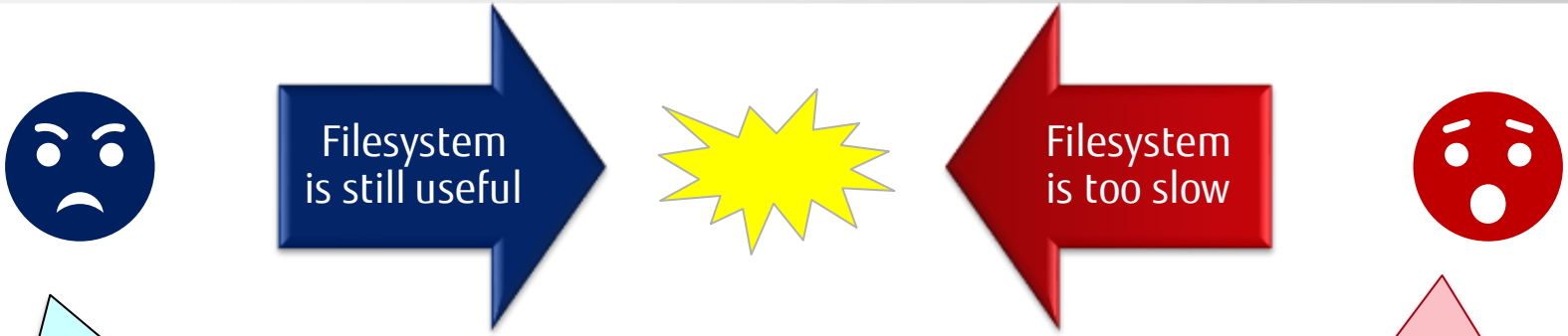
- If the data is broken, software need to detect it and correct it

Need data area management

- Software need to assign not only free area, but also used area for reuse its data
- In addition, kernel must assign the area to suitable process with authority check

Confliction of requirements

FUJITSU

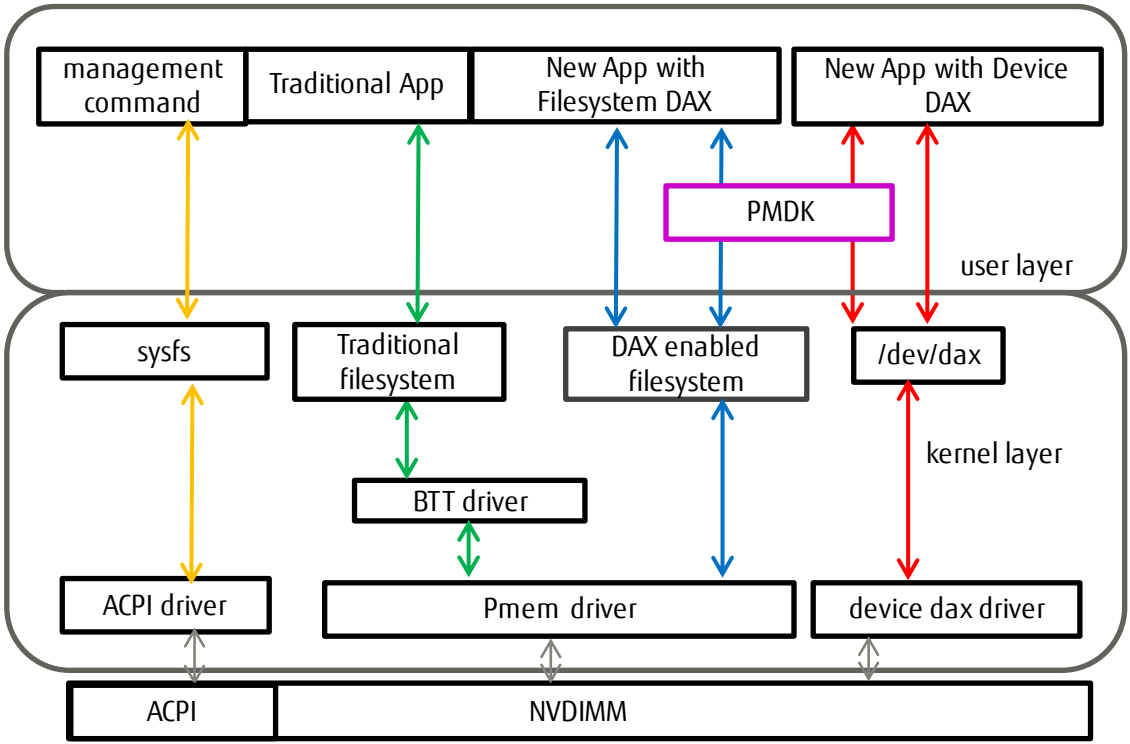


- Filesystem gives many solution for the previous considerations
  - Format compatibility of filesystem
  - Data correction
  - Journaling, or CoW
  - region management
  - authority check
  - etc...
- It is useful for current software

- Traditional I/O stack is too heavy
  - call system call
  - page cache
  - block device layer
  - etc...
- CPU cache flush is enough to make persistency
- Need new access interface to NVDIMM for next generation software

Interfaces of NVDIMM (1/3)

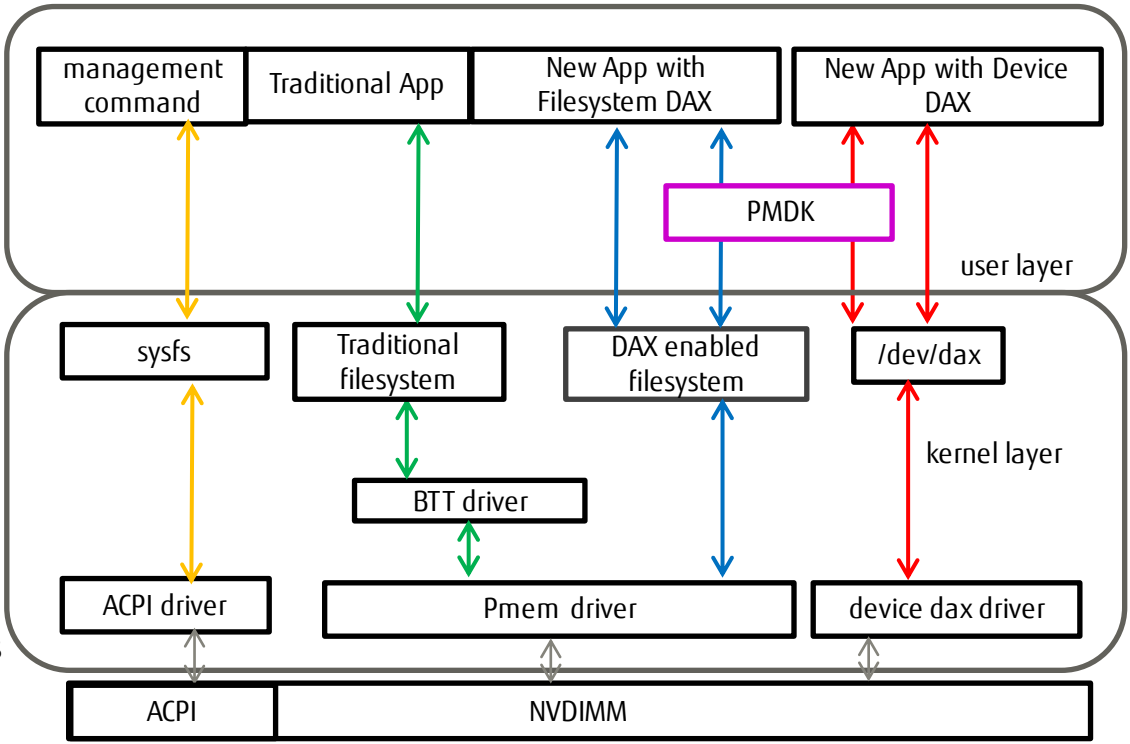
- Because of the previous reasons, Linux provides some interfaces for application
  - Storage Access(green)
    - Application can access NVDIMM with traditional I/O IF like SSD/HDD
    - So, application can use this mode without any modification
  - Filesystem DAX(blue)
    - Page cache is skipped when you use read()/write() on Filesystem-DAX
    - Application can access NVDIMM area directly if it calls mmap() for a file
    - Need filesystem support
      - Xfs, ext4...
    - This mode is suitable for modifying current applications for NVDIMM.



Interfaces of NVDIMM (2/3)

FUJITSU

- Because of the previous reasons, Linux provides some interfaces for application
  - Device DAX(red)
    - Application can access NVDIMM area directly if it calls mmap() for /dev/dax
    - /dev/dax allows only open(), mmap(), and close()
      - IOW, you can not use read()/write() nor any other system call
    - For innovative new application with NVDIMM
  - PMDK(purple) is provided
    - Set of convenient libraries and tools for filesystem DAX and Device DAX
      - Transaction support for pmem applications
      - Pool management in the DAX file/device
      - Etc.



Interfaces of NVDIMM(3/3)

FUJITSU

- NVDIMM is shown as device files like storage
  - For Storage Access : /dev/pmem##s (# means number)
  - Filesystem DAX: /dev/pmem##
  - Device DAX: /dev/dax##.#
- ndctl(*) can create these device when it creates namespace
  - Example of Filesystem DAX
  - You can make filesystem on /dev/pmem##s or /dev/pmem##

```
$ sudo ndctl create-namespace -e "namespace0.0" -m fsdax -f
{
  "dev":"namespace0.0",
  "mode":"memory",
  "size":"5.90 GiB (6.34 GB)",
  "uuid":"dc47d0d7-7d8f-473e-9db4-1c2e473dbc8f",
  "blockdev":"pmem0",
  "numa_node":0
}
$ sudo ls /dev/pmem*
/dev/pmem0
```

(*) a set of tools/commands for NVDIMM

- Note: /dev/dax##.# is character device
  - Since you cannot use read()/write() for /dev/dax##.#, you cannot use dd command for backup
  - You need to daxio command of PMDK instead of it

Filesystem-DAX is still experimental status

FUJITSU

Filesystem-DAX is very expected interface...

- The management way of NVDIMM is almost same with traditional filesystem
  - Operator can use traditional command to manage NVDIMM area
- Not only application can access NVDIMM area directly, but also it can use traditional system call

  - In contrast, Device DAX requires pool management by tools of PMDK
    - Otherwise, a software need to posses whole of the namespace (/dev/dax)
  - In addition, Application can NOT use many system call in Device DAX

But it is still experimental....

- The "experimental" message is shown when the filesystem is mounted with DAX option
  - There are difficult issues in kernel layer for some years

What is the reason?

FUJITSU

Issues of Filesystem-DAX

- What is solved, and what is current issues

Why Filesystem-DAX is experimental?

FUJITSU

- In summary, there are 2 big reasons

Filesystem DAX combines storage and memory characteristics

- This causes corner-case issues of Filesystem-DAX
- They are often difficult problem

More additional features were required, but they are/were difficult to make

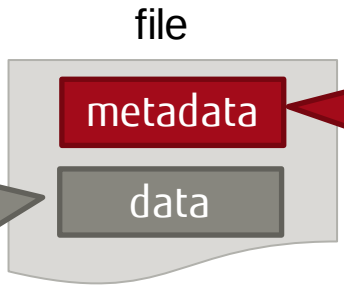
- Configure DAX on/off for each inode (directory or file)
- Co-existence with CoW filesystem

Corner Case Issue 1 : Update metadata(1/3)

FUJITSU

The first problem is “updating metadata” of the file

In Filesystem-DAX, we expected that application can make persistency of data with only cpu cache flush as I said....



However, this also means there is no chance to update metadata by kernel/filesystem

- Update time of the file may not be correct
- If an application use write some data to file on the filesystem DAX, and a user remove some blocks of the file by truncate(2), kernel cannot negotiate it
  - Data of the file may be lost
- If data transferred by DMA/RDMA to the page which is allocated as filesystem DAX, similar problem may occur

Corner Case Issue 1 : Update metadata(2/3)

FUJITSU

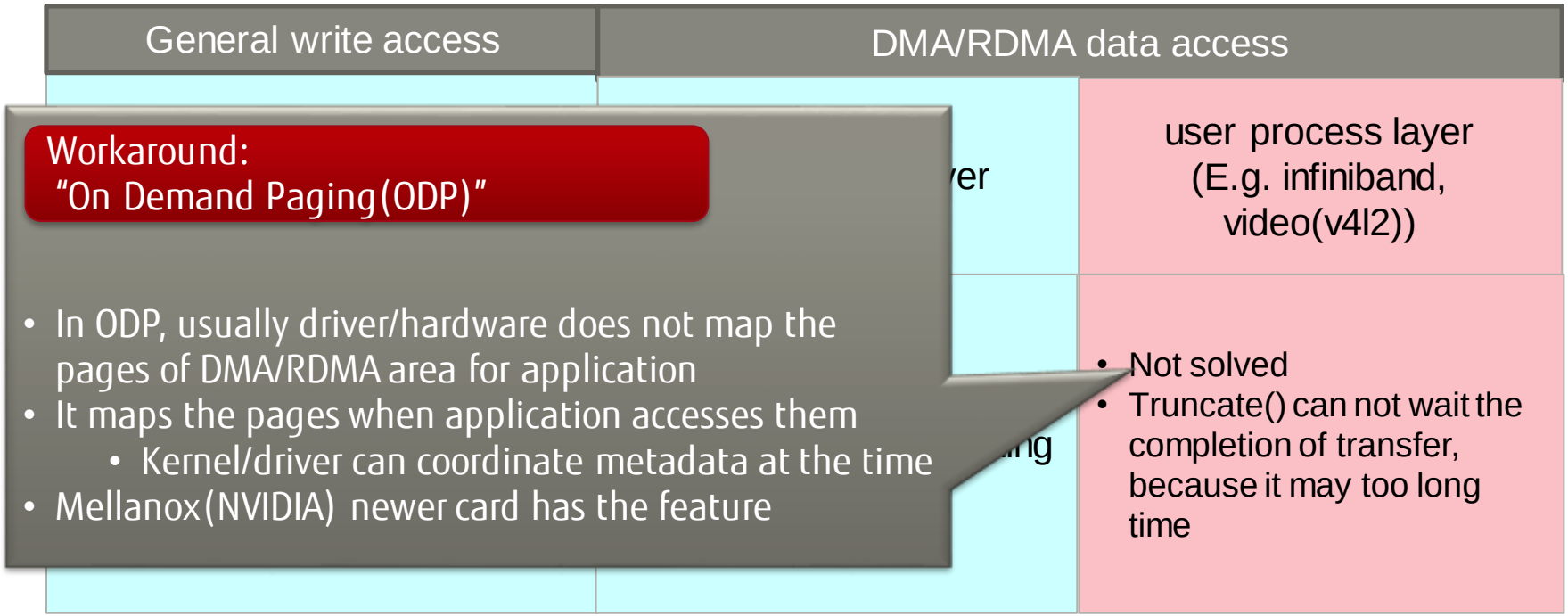
Current Status of update metadata problems

<table>
<tr><th>General write access</th><th colspan="2">DMA/RDMA data access</th></tr>
<tr><td rowspan="2">• Solved by introducing new MAP_SYNC flag of mmap()
  • Page fault occurs every write access, then kernel can update meta data
  • PMDK specifies this flag</td><td>Kernel/driver layer</td><td>user process layer
(E.g. infiniband, video(v4l2))</td></tr>
<tr><td>• Solved by waiting truncate() until finishing RMDA</td><td>• Not solved
• Truncate() can not wait the completion of transfer, because it may too long time</td></tr>
</table>

Corner Case Issue 1 : Update metadata(3/3)

FUJITSU

Current Status of update metadata problems



Corner Case Issue 2: unbind(1/2)

FUJITSU

- Unbind is a sysfs interface to disconnect / hot-remove a device
  - Each device driver provides its handler for it
  - Though NVDIMM is not hotplug device physically, its interface can be used to disable and switch the mode of NVDIMM namespace
    - Ex)
      - To change namespace mode from Filesystem-DAX to Device-DAX
      - To allow that users can NVDIMM like normal RAM
      - Etc.

```
Example of “how to use Device DAX namespace like a normal RAM”
1) Remove the device from  “Device DAX” infrastructure
# echo -n dax0.0 > /sys/bus/dax/drivers/device_dax/remove_id
# echo -n dax0.0 > /sys/bus/dax/drivers/device_dax/unbind

2) Bind it to kmem driver
# echo -n dax0.0 > /sys/bus/dax/drivers/kmem/new_id
# echo -n dax0.0 > /sys/bus/dax/drivers/kmem/bind
```

Corner Case Issue 2: unbind(2/2)

FUJITSU

- Unbind is likely “surprising remove” interface
  - There is no way to fail of unbind even if a user is using it
    - It must be disabled forcibly

      A race condition was reported between Filesystem-dax and unbind in 2021/Feb

      https://lore.kernel.org/linux-btrfs/CAPcyv4g3ZwbdLFx8bqMcNvXyrob8y6sBXXu=xPTmTY0VSk5HCw@mail.gmail.com/
- To solve this problem, Filesystem-dax needs to disable a range of NVDIMM area immediately
  - Currently, this is not solved yet
  - It will be solved after the end of Ruan's work which will be talked by him today
    - His new code will help to solve it

DAX on/off for each inode (directory or file) (1/3)

FUJITSU

Expected use-cases

Need more fine-grain settings

Users may want to change the DAX mode depending on each file

Change DAX attribute by application

Configuration is always painful for administrator
If application can detect and change it, it will be helpful for them

Performance tuning

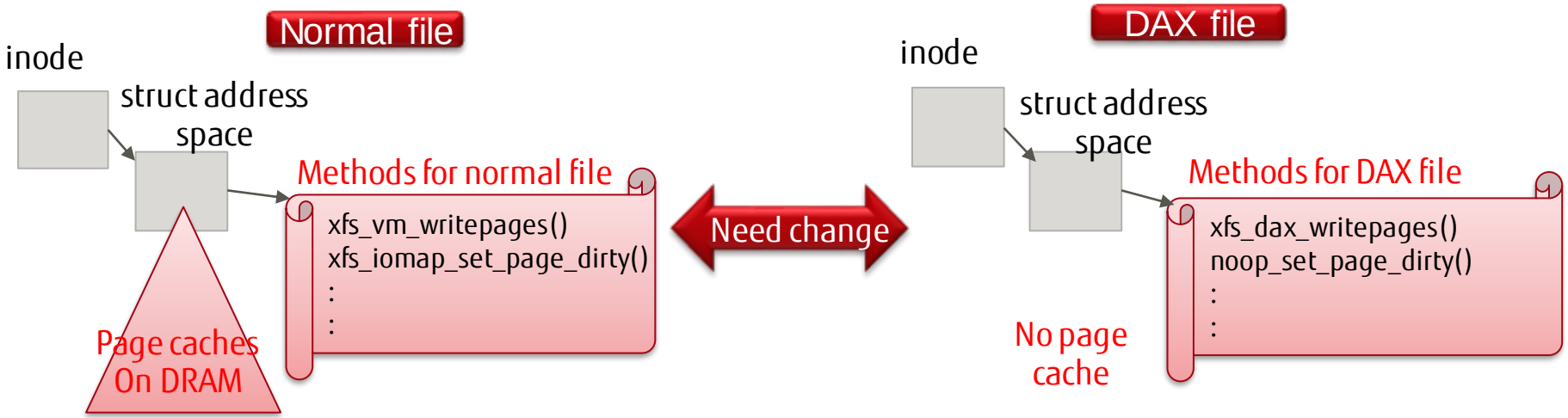
Since the write latency of NVDIMM is a bit slower than RAM,
User may want to use page cache by DAX off setting

Workaround when Filesystem-DAX has a bug

DAX on/off for each inode (directory or file) (2/3)

FUJITSU

- What was the problem
  - If filesystem change DAX attribute, filesystem need to change methods of filesystem between DAX and normal file, but they may be executed yet
  - Data of the page cache must be moved silently when the dax attribute becomes off
    - These problems were very difficult

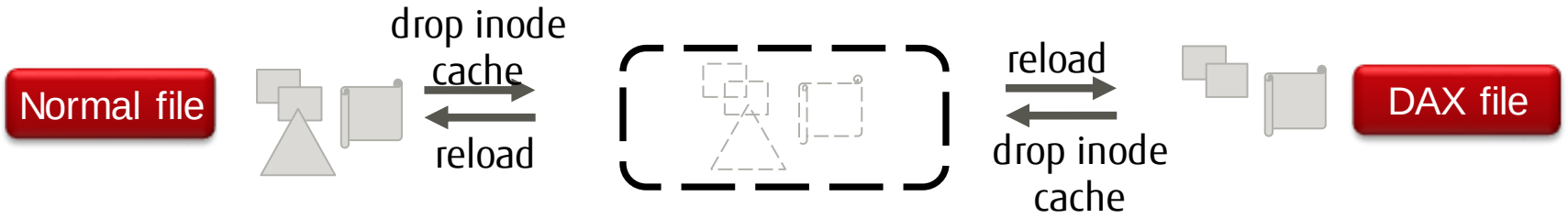


DAX on/off for each inode (directory or file) (3/3)

FUJITSU

- Fortunately, this issue was solved
  - The DAX attribute is changed only when its inode cache is NOT loaded on memory
    - Filesystem can load suitable methods for each attribute when it reloads inode to memory
    - Page caches of the file are also dropped
    - Users can use this feature with the new mount option. #mount ... -o dax=inode
    - The DAX attribute is changed by command

```
DAX on : $xfs_io -c 'chattr +x' <file or directory>
DAX off: $xfs_io -c 'chattr -x' <file or directory>
```

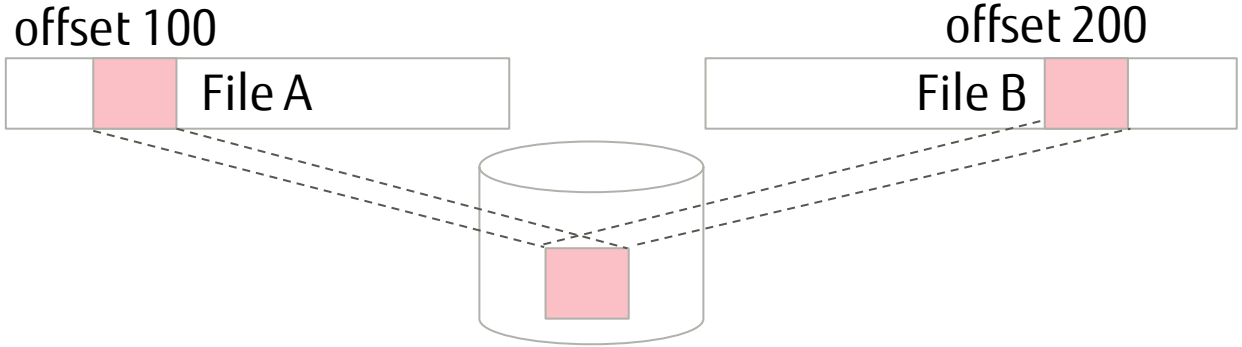


  - Note
    - All of applications which use the target file must close it to change the dax attribute
      - Filesystem will postpone changing the DAX attribute until dropping inode cache and page cache of the file

Coexisting with CoW (reflink/dedup) filesystem (1/2)

FUJITSU

- The Copy on Write feature of filesystem (xfs:reflink/dedup, btrfs)
  - If there is a same data block on different files, filesystem can merge it as same block



  - So far, if only filesystem manages such block, it was enough
    - Since a page cache is allocated for each file of the block, memory management layer don't need to know it
  - In Filesystem-DAX, it becomes problems
    - Merged block equals merged memory itself, it affects the memory failure case

Coexisting with CoW (reflink/dedup) filesystem (2/2)

FUJITSU

Problems

Need actual CoW implementation for Filesystem DAX

- Currently, there is no implementation of reflink/dedupe for Filesystem-DAX
  - Iomap, which is newer io block layer instead of buffer_head, has interface for CoW filesystem
  - XFS filesystem DAX also uses iomap
  - But there is no code to use CoW and DAX at the same time

Needs to chase plural files from a merged page/block

- When a memory failure occurs, need to kill processes which use the memory
- To achieve it, kernel need to find all processes from the merged page/block
  - But a merged page has only one struct page
  - No space for plural files in it

Ruan-san will talk how to solve them

FUJITSU

Deep dive to solve the issue of Filesystem-DAX

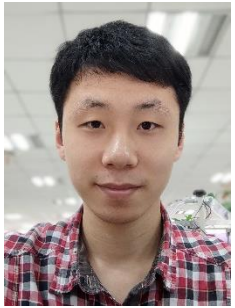
- Support reflink/dedupe for FSDAX*
- Improve NVDIMM-based Reverse mapping

* Filesystem-DAX

Self-introduction

FUJITSU

- Ruan Shiyang
  - A Software Engineer of Fujitsu Nanda
  - Experience in Embedded development
  - Currently focusing on Linux filesystem and persistent memory



Background

FUJITSU

- fsdax is “EXPERIMENTAL”
  - reflink and fsdax cannot work together

Create a XFS (reflink is enabled by default) and mount it with “dax” option

```
$ mkfs.xfs /dev/pmem0 && mount -o dax /dev/pmem0 /mnt
```

Then error occurs

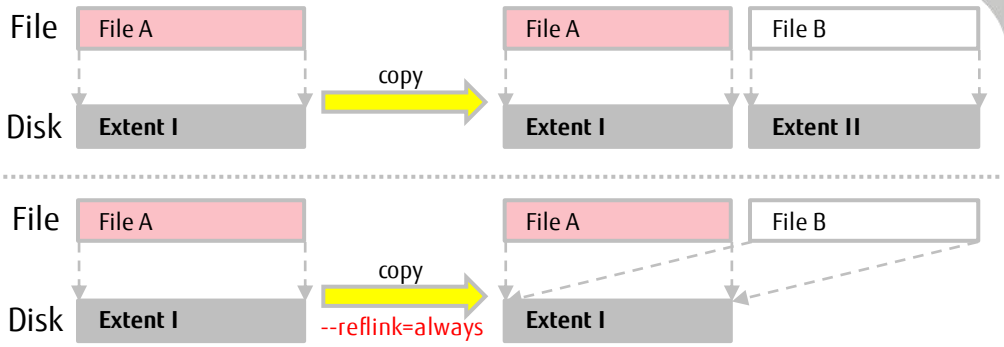
```
mount: /mnt: wrong fs type, bad option, bad superblock on /dev/pmem0,
missing codepage or helper program, or other error.
```

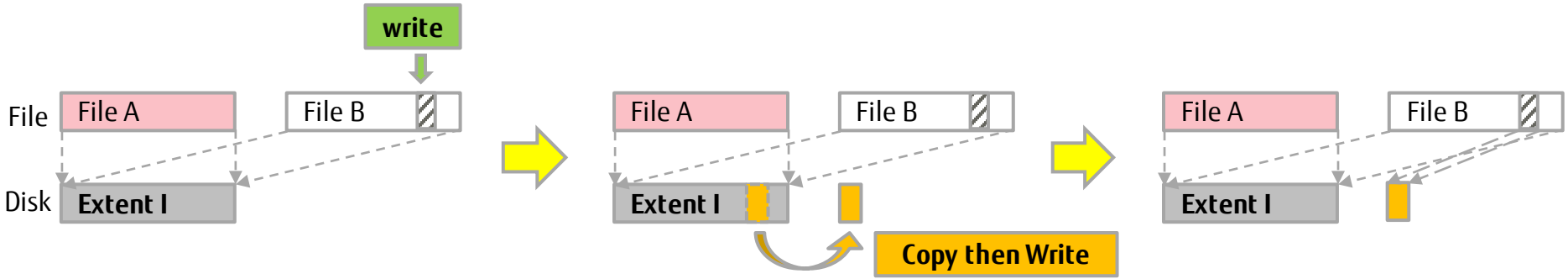
The dmesg shows the reason

```
XFS (pmem0): DAX enabled. Warning: EXPERIMENTAL, use at your own risk
XFS (pmem0): DAX and reflink cannot be used together!
```

reflink

- Files share extents for same data
- Advantages
  - Fast copy
  - Save storage
- Copy on Write mechanism (CoW)
  - Copy the shared extents before writing data

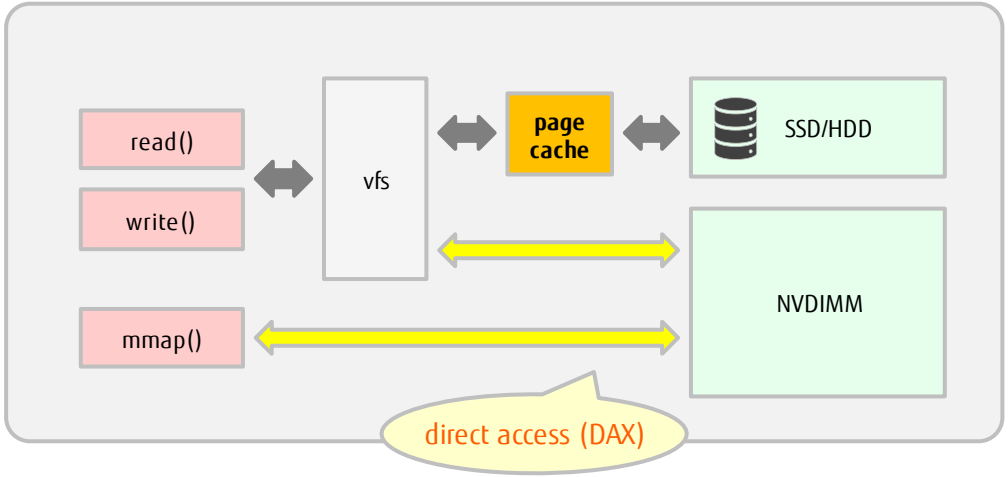




fsdax (Filesystem DAX)

FUJITSU

- A mode of a NVDIMM namespace
- No **page cache** in I/O path
- Allows **direct mappings** to persistent memory media



Why **reflink** and **fsdax** cannot work together?

FUJITSU

Issues

1. Support reflink/dedupe for FSDAX
   - Extent iomap interface
   - Support CoW in fsdax
   - Support Dedupe in fsdax
2. Improve NVDIMM-based Reverse mapping
   - Support 1-to-N Reversed mapping for NVDIMM

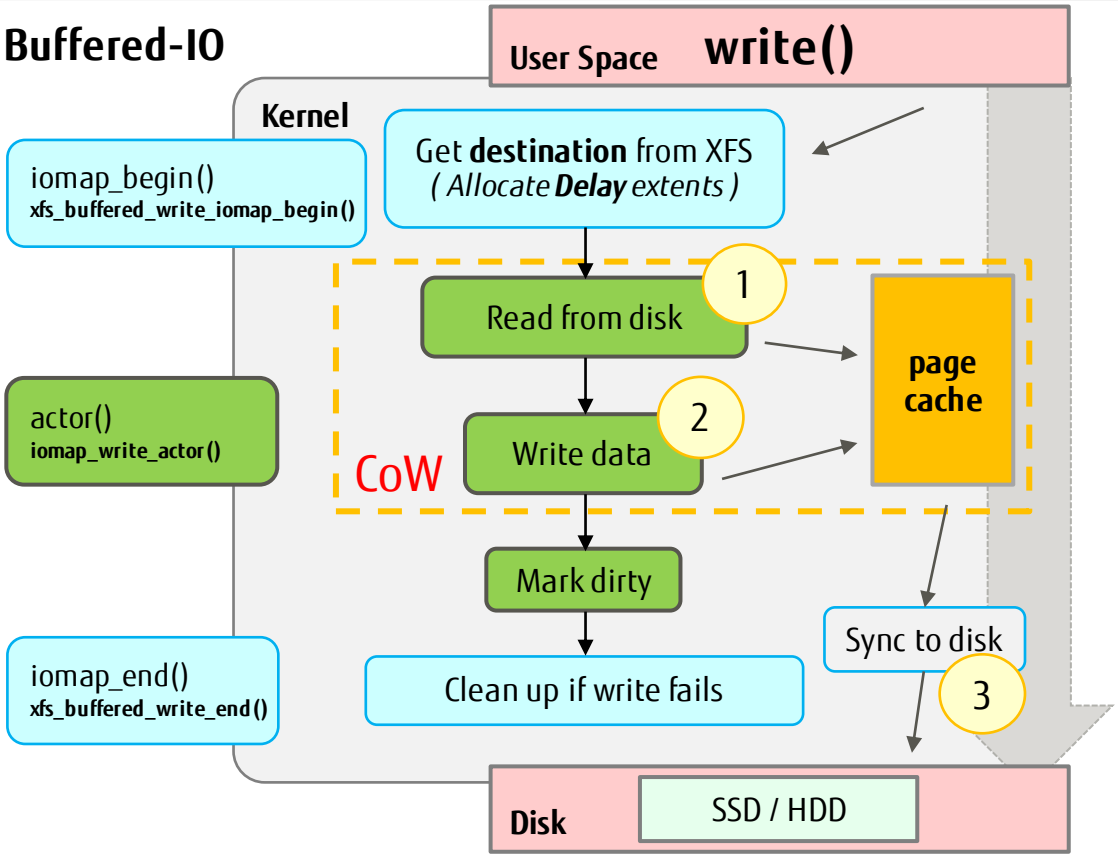
FUJITSU

Support reflink/dedupe for FSDAX

- Difference between Buffered-IO vs. FSDAX
- What must be implemented?
- How are they implemented?

Comparison: Buffered-IO vs. FSDAX write() (1/2)

FUJITSU



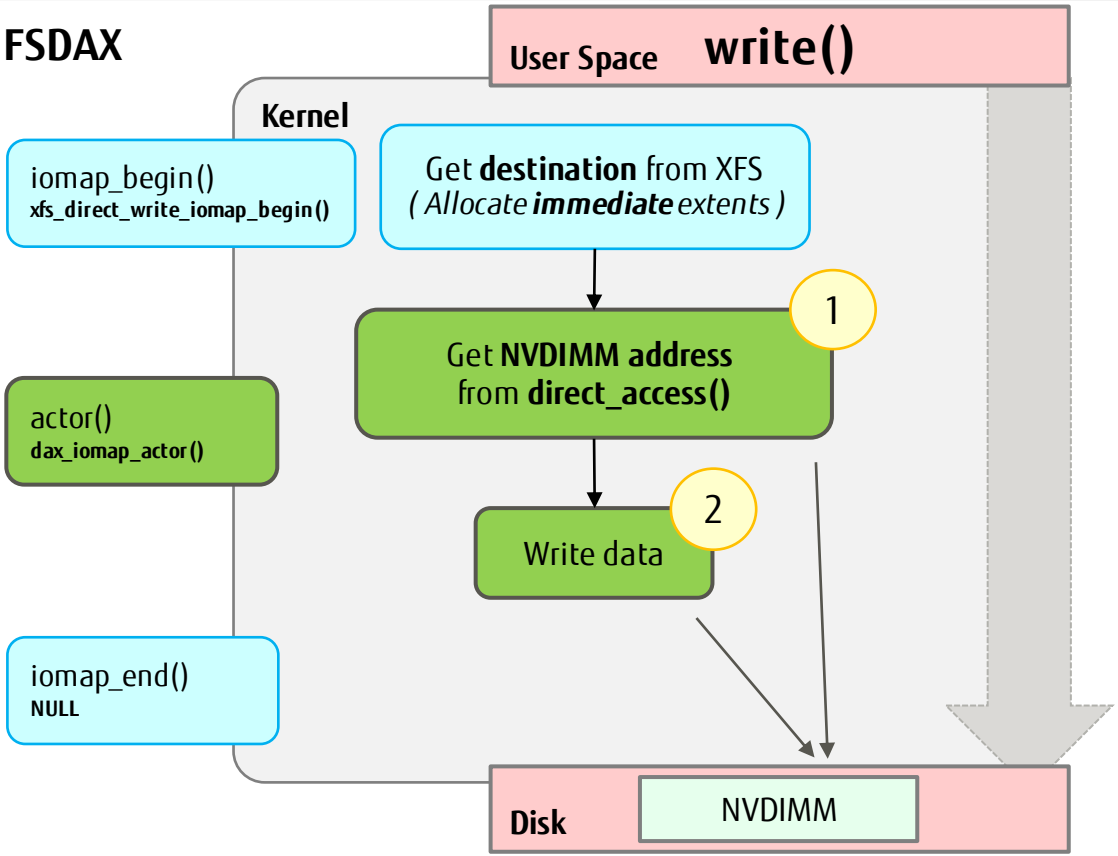
- Requires page cache

1. Read (Copy) *destination data*
   - from *disk*
   - to *page cache*
2. Write *user data*
   - from userspace
   - to *page cache*
3. Sync *page cache* to *disk*
   - remap CoW extents

*The progress of using page cache indicate the **CoW** operation.*

Comparison: Buffered-IO vs. FSDAX write() (2/2)

FUJITSU



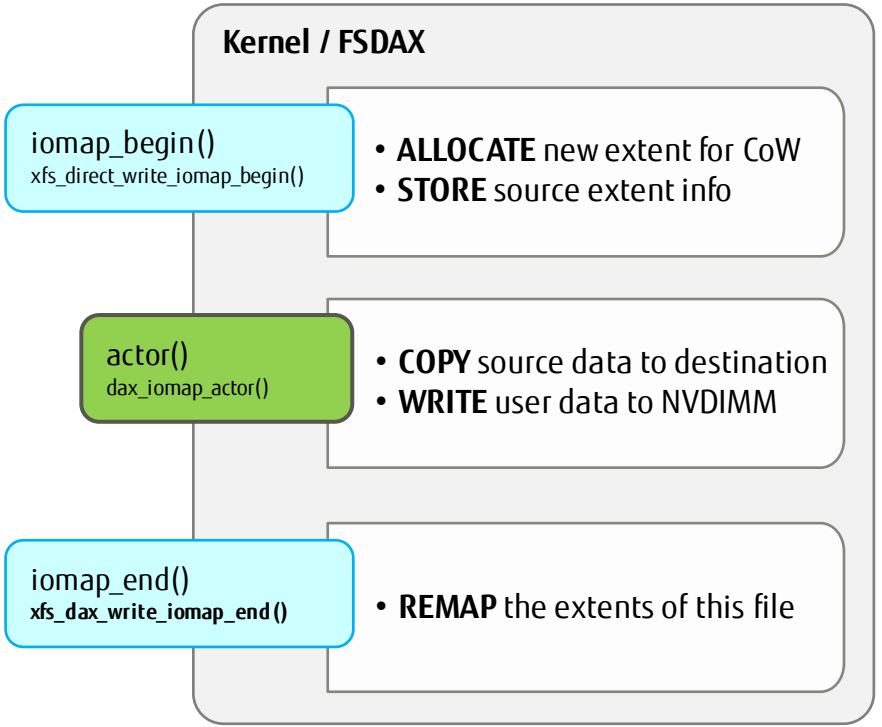
- No page cache
  - directly access to NVDIMM

1. Not Copy
   - get *NVDIMM address* only
2. Directly Write
   - *user data* to *NVDIMM*
   - no need to sync
   - cannot remap extents

CoW is missing

What must be implemented?

- Extent iomap interface
  - Introduce 'srcmap'
  - Fill 'iomap' & 'srcmap' for fsdax
- Support CoW in fsdax
  - Add CoW for write()
  - Add CoW for mmap()
  - Remap extents after CoW
- Support Dedupe in fsdax
  - Add a 'dax' deduplication

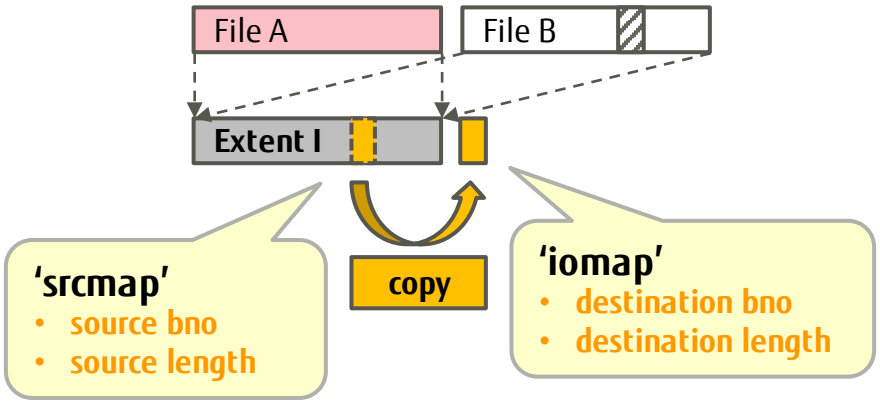


Enhancement in iomap framework

iomap interface - Introduce 'srcmap'

FUJITSU

- What is necessary
  - Require **source** info for CoW
    - Only use 'iomap' to tell **destination** to write
    - but CoW requires more info
      - **source** bno: where to copy from
      - **source** length: how much to copy
- How?
  - Introduce another struct called '**srcmap**'
    - remember & pass the source info

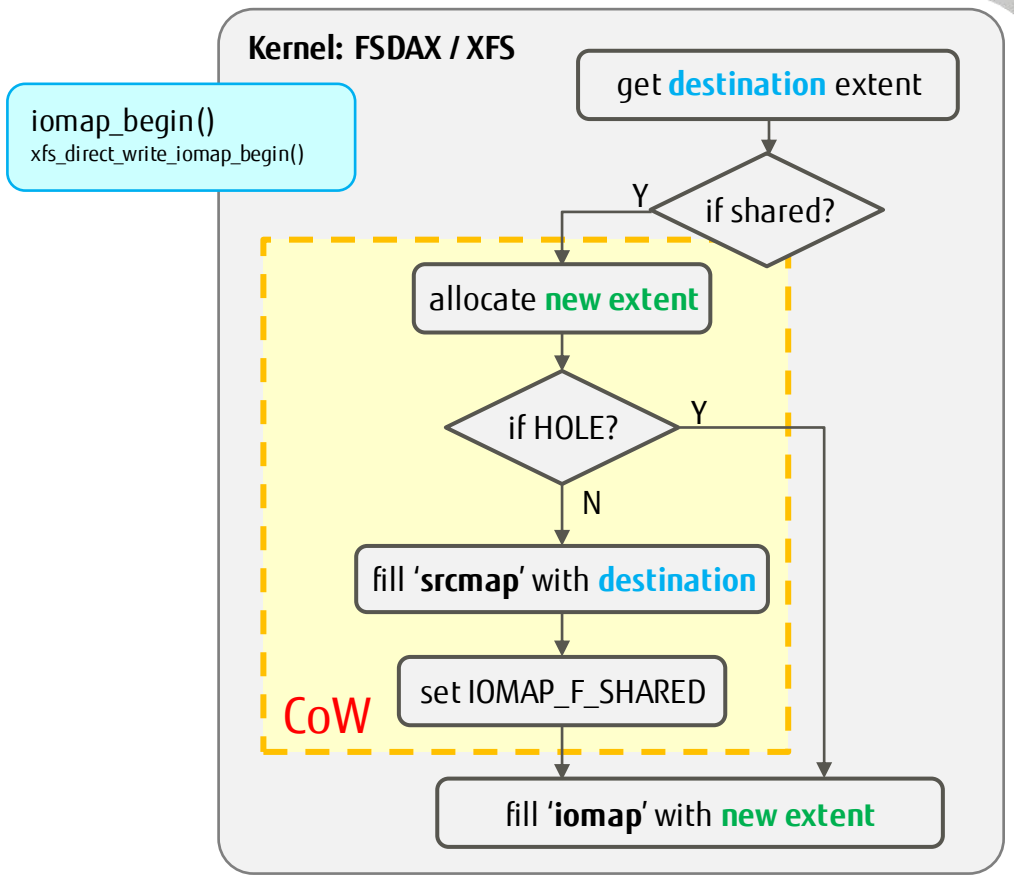


* Introduced by Goldwyn Rodrigues: [PATCH] iomap: use a srcmap for a read-modify-write I/O

iomap interface – Fill 'iomap' & 'srcmap'

FUJITSU

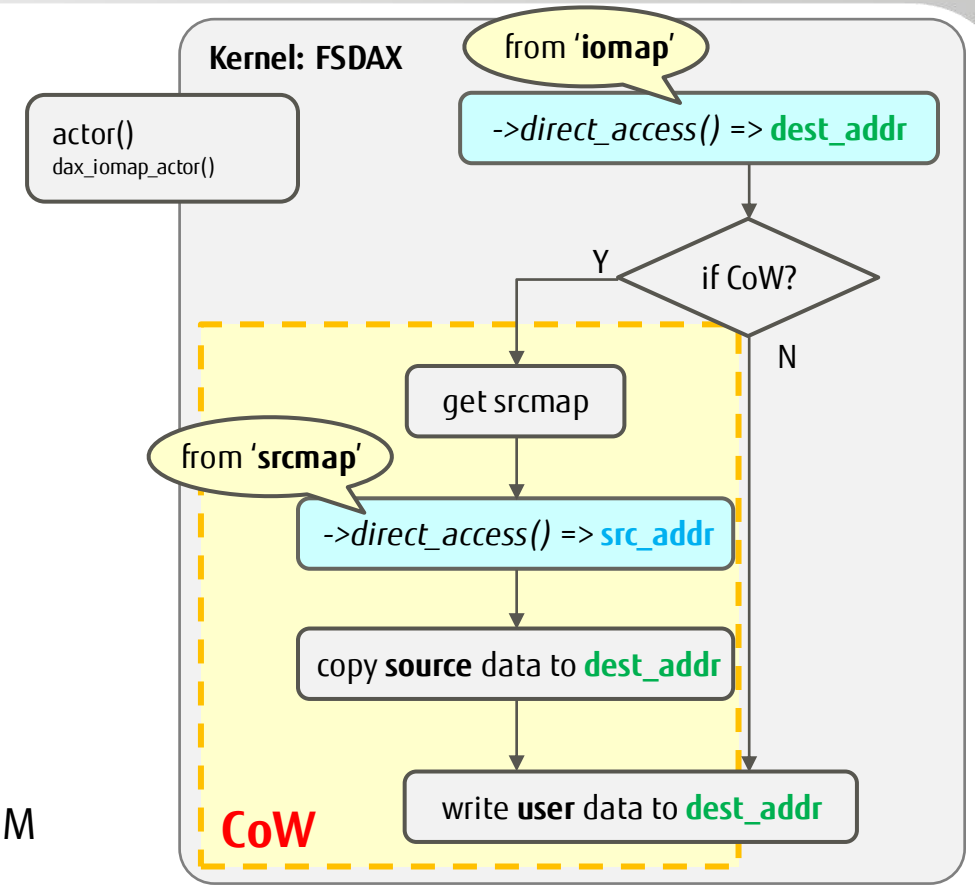
- What is necessary
  - Store source info in '**srcmap**'
    - Only fill 'iomap' in *->iomap_begin()*
      - Implemented by filesystem
    - Need to fill 'srcmap' for shared extent
- How?
  - Add CoW branch
    - Allocate new extent for CoW
    - Fill 'srcmap' with destination extent
    - Fill 'iomap' with new extent
    - Set IOMAP_F_SHARED flag



Support CoW - Add CoW for write()

FUJITSU

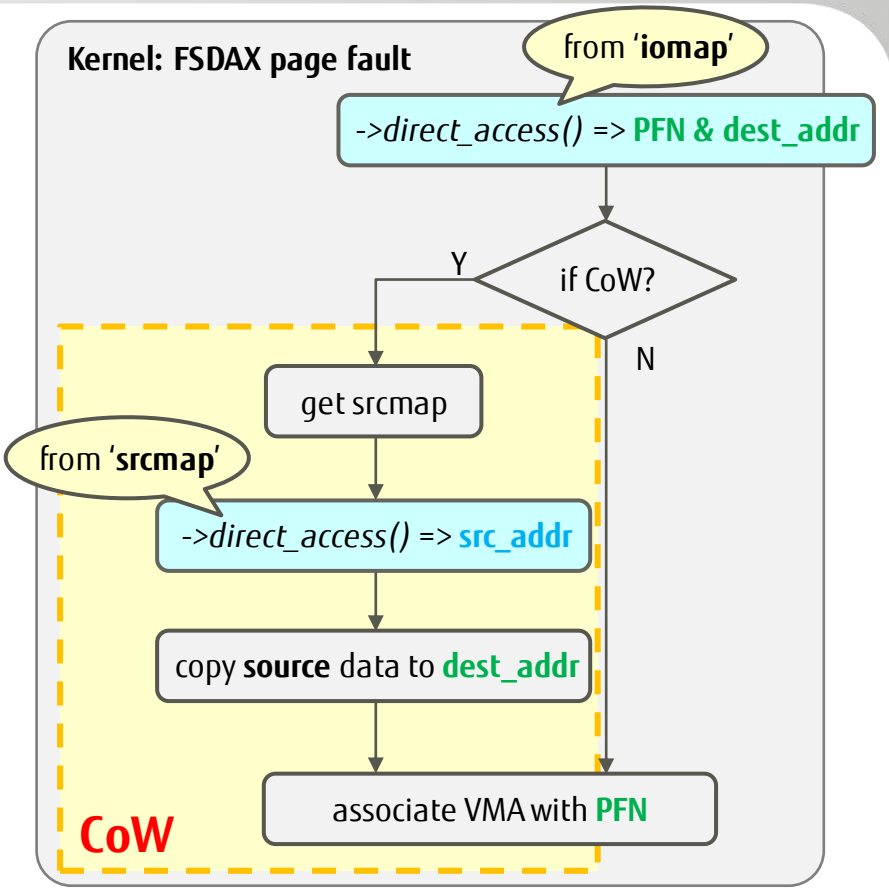
- What is necessary
  - Execute CoW in **write()** path
    - Only write user data to destination
      - Use *->direct_access()* to translate address
        - *From:* offset in block device
        - *To:* physical memory address in NVDIMM
    - Need a pre-copy before writing
      - *src_addr*: (translated) source address
      - *dest_addr*: (translated) destination address
- How?
  - Add CoW branch
    - Copy source data from '**srcmap**' to NVDIMM



Support CoW - Add CoW for mmap()

FUJITSU

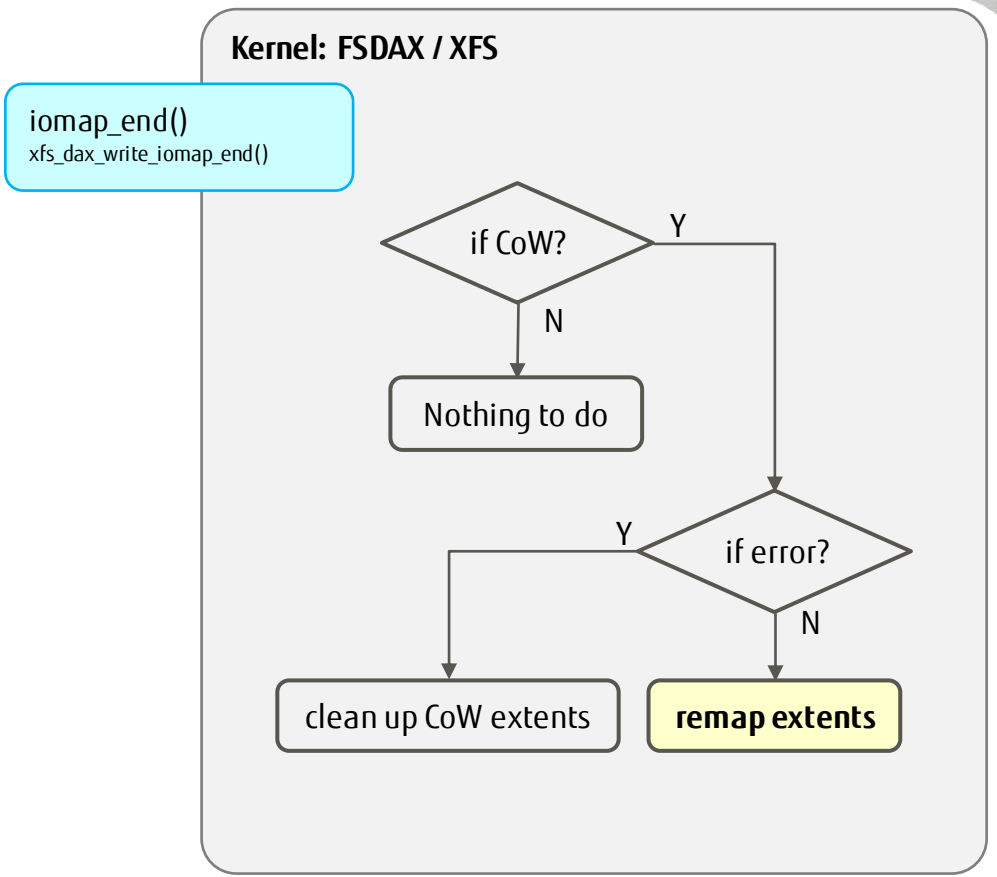
- What is necessary
  - Execute CoW in **page fault**
    - FSDAX has its own specific PTE&PMD fault
      - Use iomap framework
      - Only find destination PFN & associate VMA
    - Need a pre-copy before associating
      - *PFN*: destination PFN found by *->direct_access()*
      - *src_addr*: (translated) source address
      - *dest_addr*: (translated) destination address
- How?
  - Add CoW branch in PTE&PMD fault
    - Copy source data before VMA is associated
      - Prepare for writing user data in userspace



Support CoW - Remap extents after CoW

FUJITSU

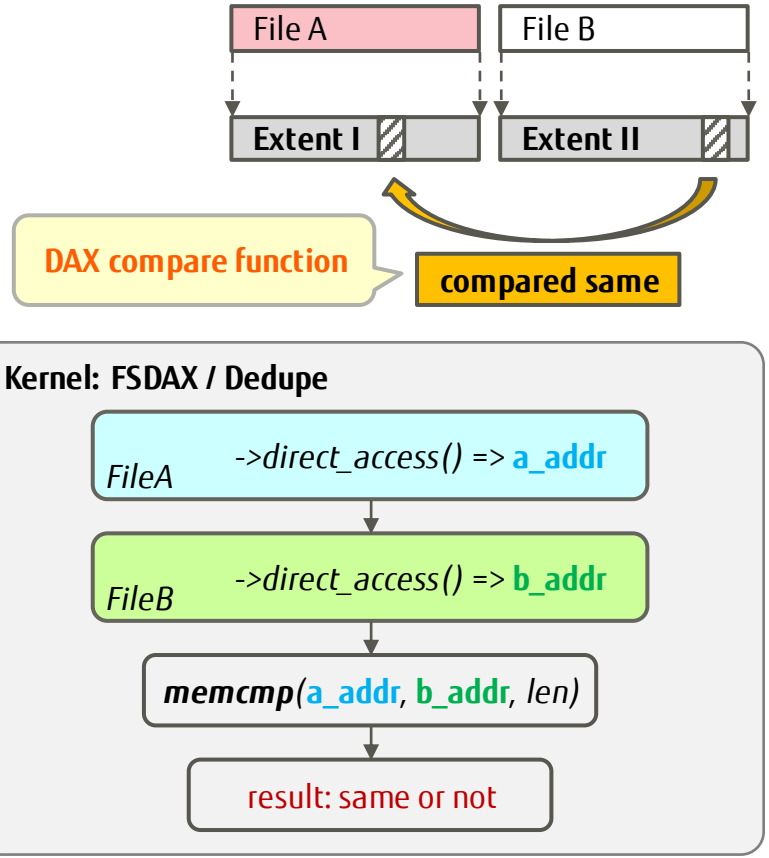
- What is necessary
  - Remap extents for CoW
    - The new extent not been mapped
      - Metadata not been updated
      - File won't contain the CoW extent
    - Need to update metadata after CoW
- How?
  - Add remap in *->iomap_end()*
    - Do remap operation if is CoW
    - Clean up if CoW operation fails



Support dedupe - Add a 'dax' deduplication

FUJITSU

- What is necessary
  - Deduplicate DAX files
    - Dedupe: reduce redundant data on storage costs
    - Require a dedupe function
      - Only have generic dedupe function
      - ➢ compare data in **page caches**
      - Not adapted to FSDAX
      - ➢ no page cache
      - ➢ directly compare data in NVDIMM
  - Need a new dax compare function
- How?
  - Introduce a DAX compare function
    - Compare data by *memcmp()*



FUJITSU

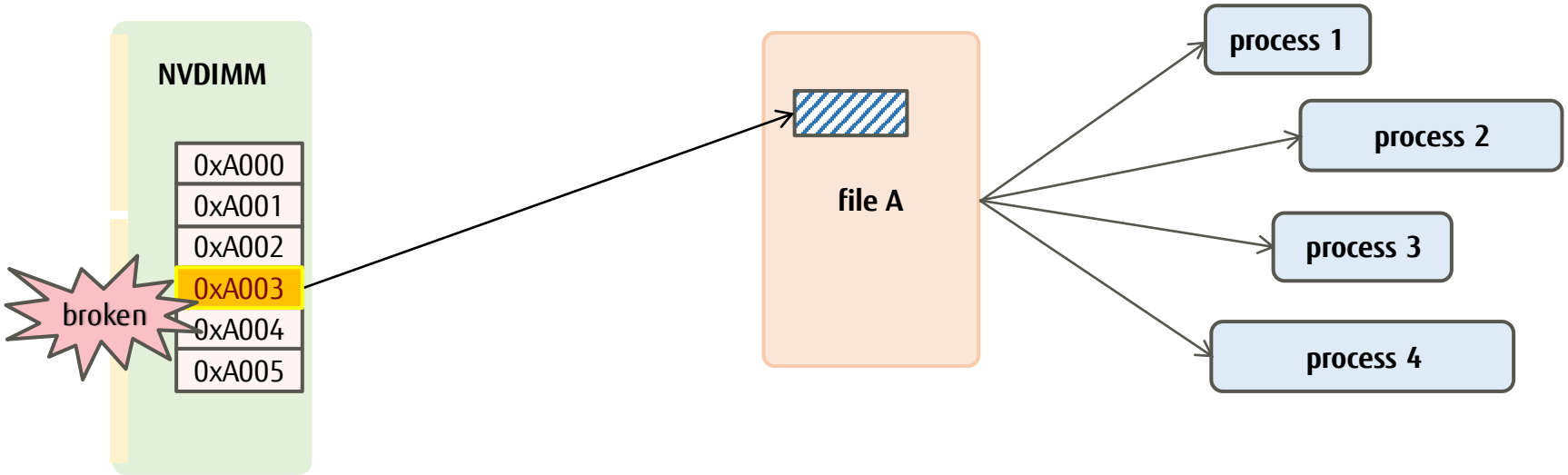
Improve the current NVDIMM-based Reverse mapping

- Why Reverse mapping needs to be improved
- Struggling to solve this issue
- What must be implemented?
- How are they implemented?

When Memory failure occurs

FUJITSU

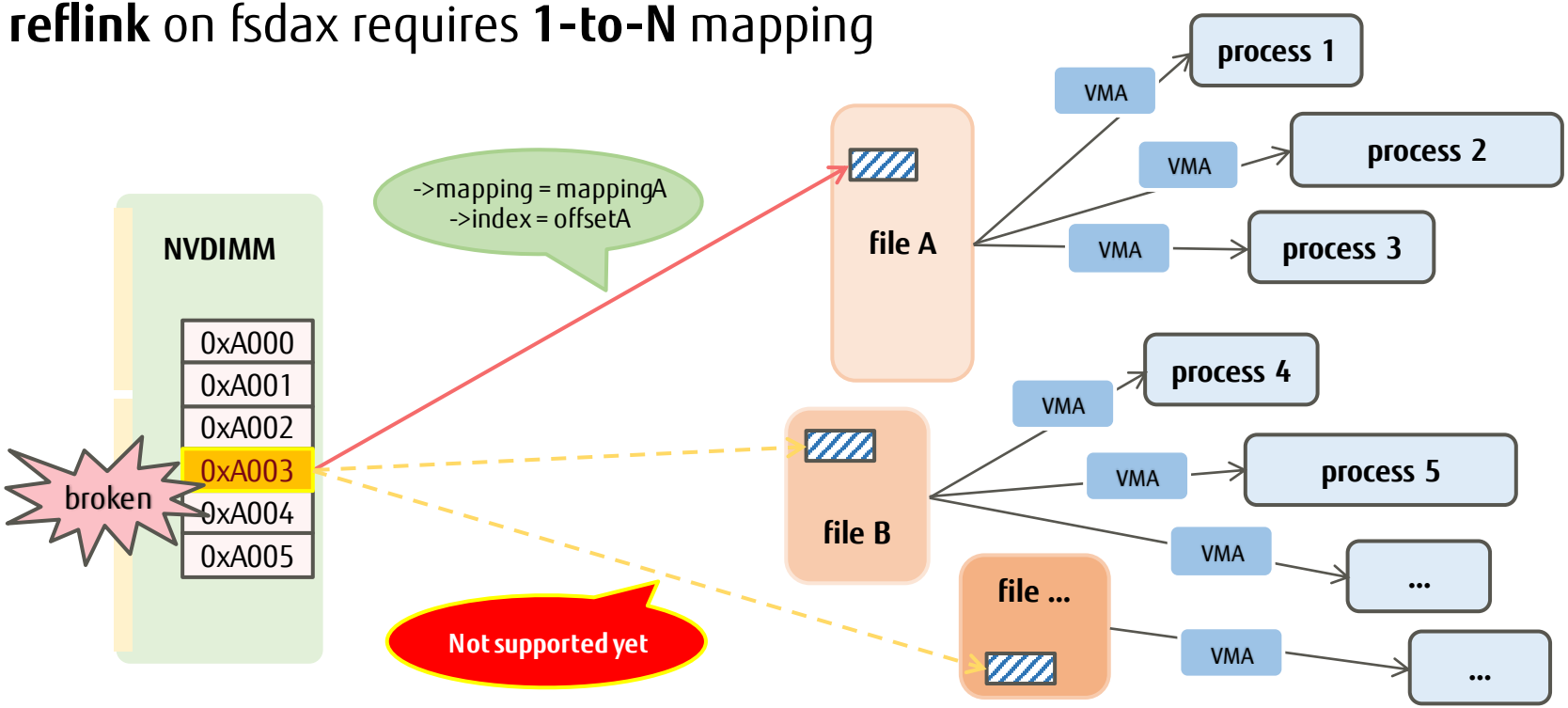
1. track all **processes** associating with the **broken page** on NVDIMM
2. send signal to kill those associated processes



Need to improve NVDIMM-based Reverse mapping

FUJITSU

- currently only support 1-to-1 mapping
- **reflink** on fsdax requires **1-to-N** mapping



Struggling to solve this issue

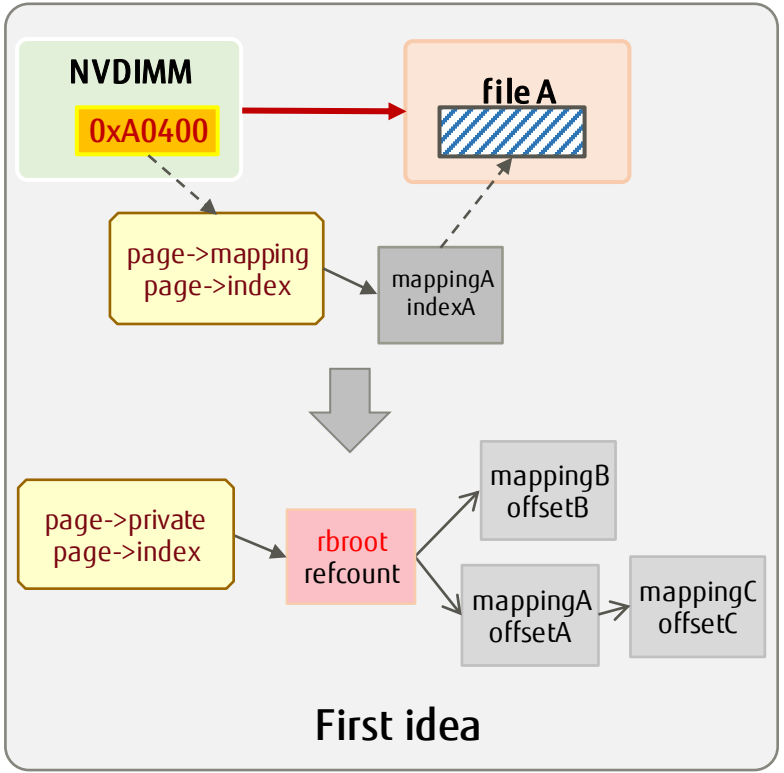
FUJITSU

- First idea was simple, but it was bad idea
  - Simply make rbtree for 1-to-N rmap

No! Cause of huge over head

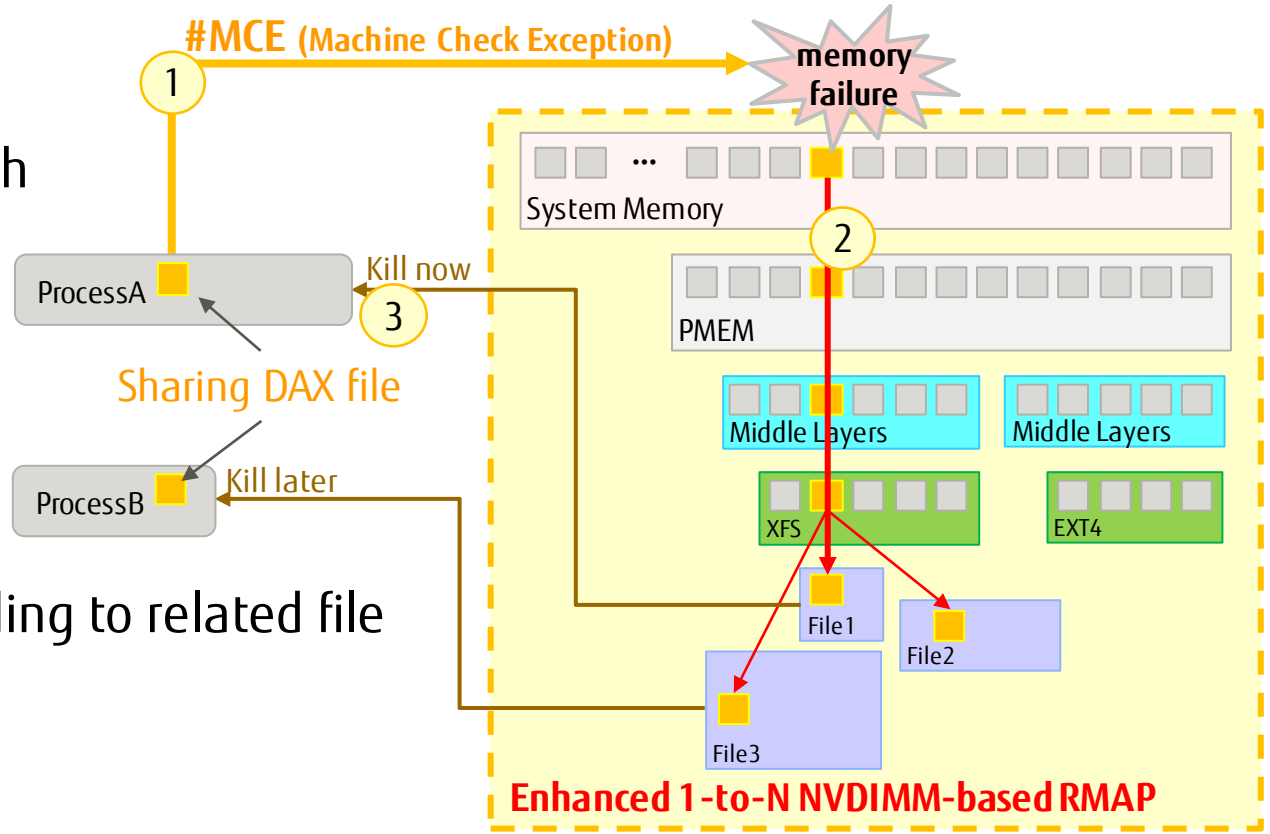
- Current strategy after some struggles
  - Chase filesystem internal to find 1-to-N relationship
  - What is difficulty of this way?
    - Memory failure information is basically page unit. But we need to find where it is in filesystem
    - Filesystem may be created on partition, and/or LVM, it affects relative offset in the filesystem

see next page



Introduce 1-to-N NVDIMM-based RMAP

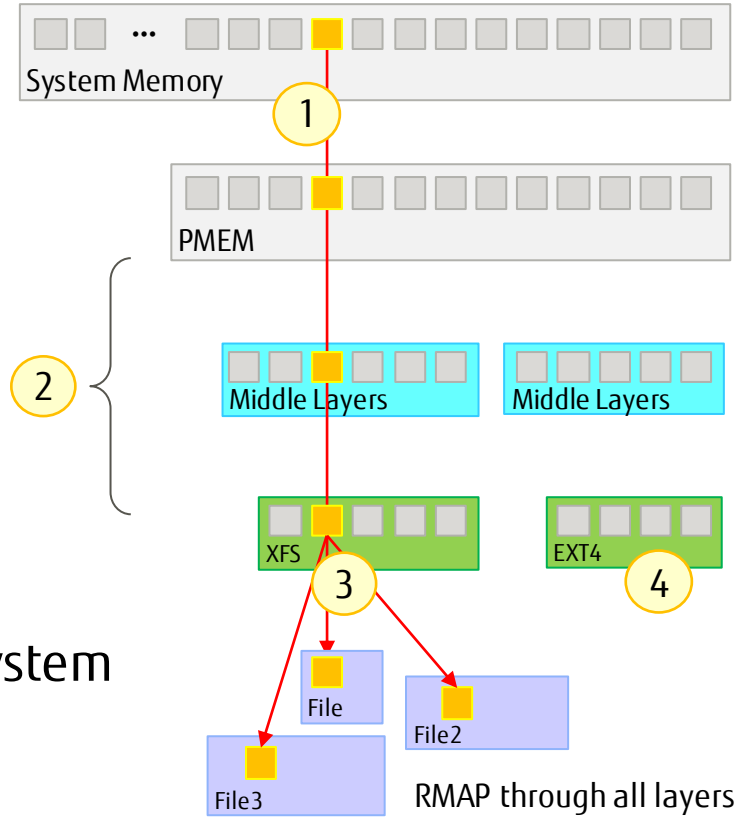
1. MCE triggers
   - memory-failure
2. 1-to-N RMAP through
   - mm layer
   - device driver
   - block device layer
   - filesystem layer
   - files
   - processes
3. Kill processes according to related file
   - Kill current immediately
   - Kill others later



What must be implemented?

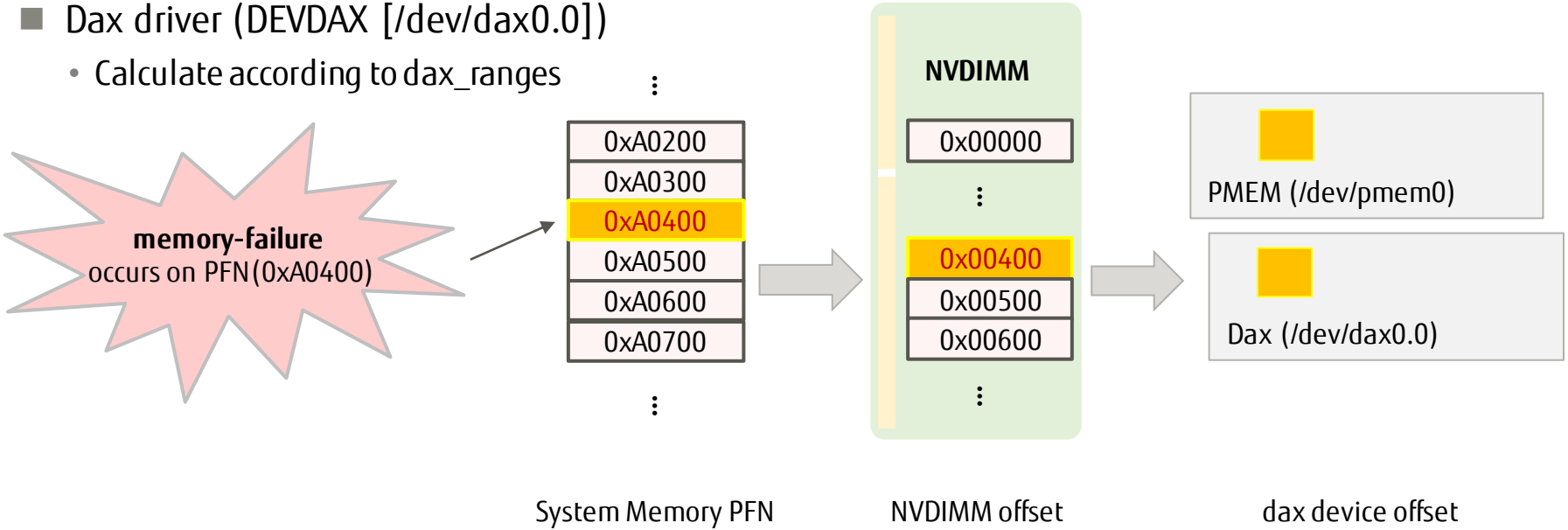
FUJITSU

1. RMAP from NVDIMM driver to dax device
2. RMAP from dax device to filesystem
   - Introduce dax_holder registration mechanism
   - Types of holder
     - Filesystem
     - Partition
     - Mapped Device
3. RMAP from filesystem to file
   - Require **rmapbt** feature
   - Improve process collection and killing for FSDAX
4. Compatibility for no-reflink / no-rmapbt filesystem
   - e.g., EXT4 does not support reflink & rmapbt



1. RMAP from **NVDIMM** driver to **dax device**

- Translate the PFN number into offset within a dax device
  - PMEM driver (FSDAX [/dev/pmem0])
    - Linear offset translation
  - Dax driver (DEVDAX [/dev/dax0.0])
    - Calculate according to dax_ranges

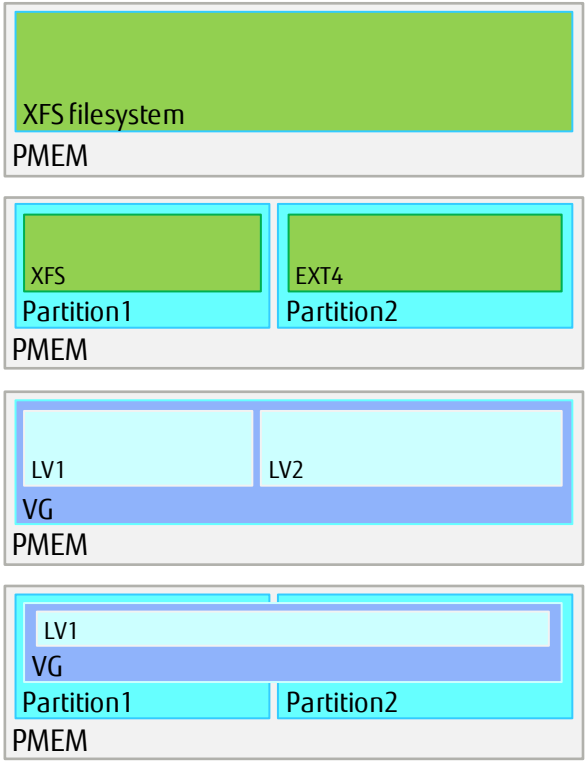


2. RMAP from **PMEM** to **filesystem** (1/3)

FUJITSU

- PMEM (FSDAX) may be used in different ways
  1. Filesystem: XFS, EXT4...
  2. Partitions
  3. Mapped Device: LVM...
  4. Nested by Partitions or mapped Devices

- Introduce dax_holder registration mechanism
  - The holder represents the inner layer of a PMEM
  - Register when holder being mounted / initialized
  - Interface for notifying memory-failure
    - holder_ops->notify_failure()

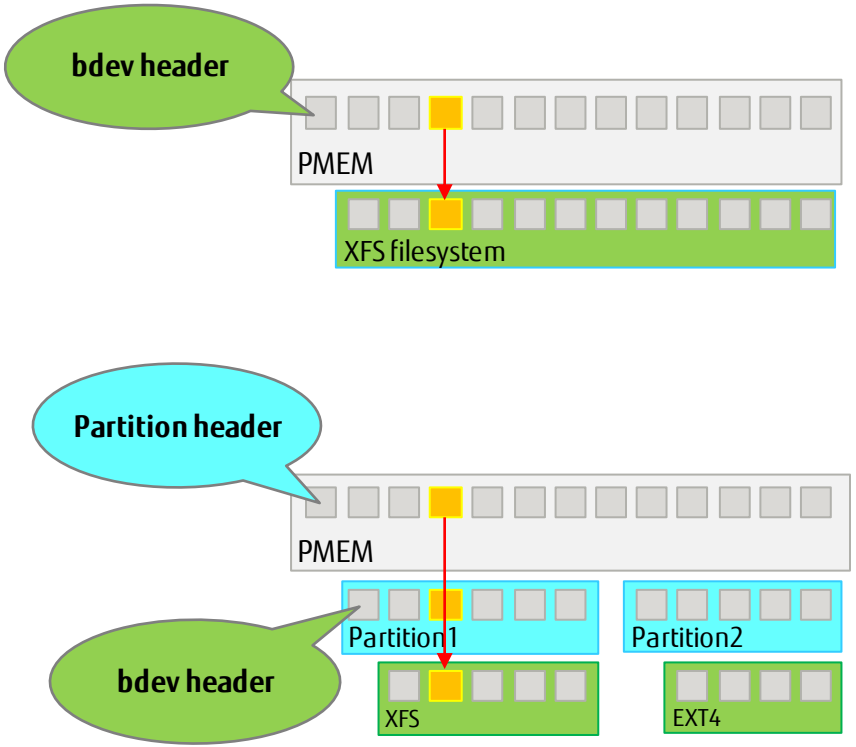


Usages of PMEM

2. RMAP from **PMEM** to **filesystem** (2/3)

FUJITSU

1. Filesystem as a holder
   - mkfs directly on a PMEM
     - `mkfs.xfs /dev/pmem0`
   - No partition in PMEM
   - Need translation
     - Remove the fixed bdev header length
2. Partition as a holder
   - Parted by tools
   - More than one partitions
   - Need translation
     - iterate each partition's range to get the location
     - remove the offset of the partition header

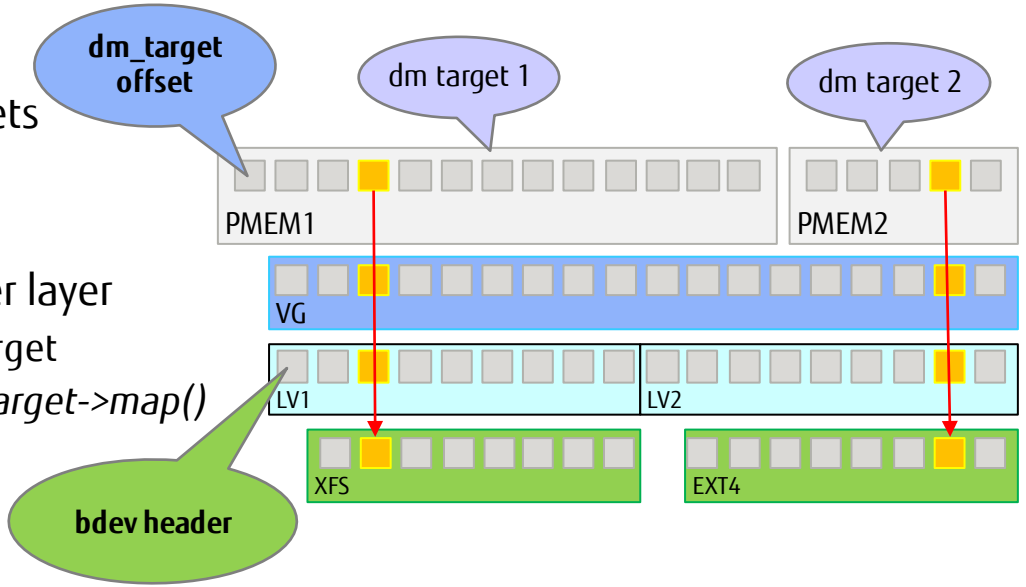


2. RMAP from **PMEM** to **filesystem** (3/3)

FUJITSU

3. Mapped device as a holder

- Created by LVM or other tools
- No partition, but one or many dm targets
- Types of dm targets
  - Linear, Raid, Crypt...
- Introduce RMAP from dm target to inner layer
  - Implement *dm_target->rmap()* in each target
  - This is reversed progress of the exist *dm_target->map()*
- Need translation
  - Iterate targets in a Mapped device
    - Find out which target is the broken PMEM
    - Remove dm_target offset for the inner layer
  - Continue RMAP in inner layer
    - The inner layer is also a holder (filesystem, partition...)



3. RMAP from **filesystem** to **file**

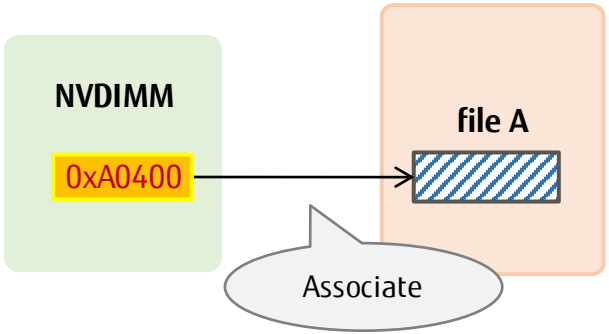
FUJITSU

- Require filesystem has '**rmapbt**' feature
  - rmapbt: Given an offset and length, search for extents contains it
  - XFS provides the query interface
    - xfs_rmap_query_range()
  - Search result could be
    - File content
      - Kill processes who are using this file
      - Try to recovery file data according to XFS log device
    - Filesystem metadata
      - Hard to recovery online, shutdown filesystem and report error
- Improve process collection and killing for FSDAX
  - The original method is page-based
    - one page indicate one file (1-to-1)
  - Need to be changed to file-based

4. Compatibility for no-reflink / no-rmapbt filesystem

FUJITSU

- Need to keep the original 1 page-to-1 file association
  - Associate page->mapping & ->offset
    - Establish the RMAP relationship with page and file
  - Not only keep the relationship, but also avoid the error
    - Currently it reports error if called more than once
    - Make it associate only once and only for the first time



- Need to keep the original RMAP routine
  - The page-based memory-failure handler still works with support of above
  - Fall back if 1-to-N RMAP routine get '-EOPNOTSUPP'

Summary of the new 1-to-N RMAP solution

FUJITSU

- 1-to-N RMAP has been implemented
  - compatible for all NVDIMM modes
  - compatible for all usage of PMEM
  - compatible for all filesystems

- What is the next?
  - Need to fix the race condition against unbind
    - With the help of 1-to-N RMAP, this can be fixed
      - My new code is basically for one page(PTE or PMD) of memory failure
      - Unbind is likely “a wide range of memory failure”, then we hope my code will help such case

FUJITSU

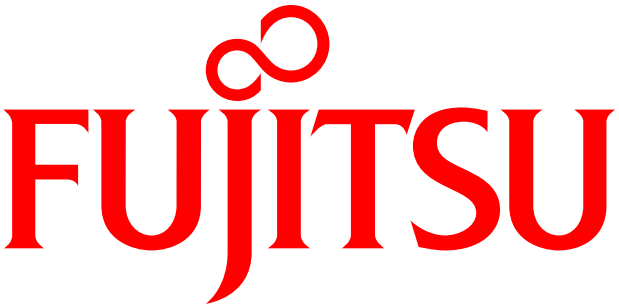
Conclusion

We talked about the followings

FUJITSU

- Basis of NVDIMM for Linux
- Issues of Filesystem-DAX (Direct Access mode)
- Deep dive to solve issues of Filesystem-DAX
  - Support reflink & dedupe for fsdax
  - Fix NVDIMM-based Reverse mapping

- Community has made many enhancement for NVDIMM on Linux
- We have worked for NVDIMM to remove experimental status of Filesystem DAX
  - We hope it will be achieved as soon as possible



shaping tomorrow with you

FUJITSU

Appendix

How to use NVDIMM in Linux

FUJITSU

- Make region
  - You can configure it on BIOS screen (or ipmctl command for Intel DCPMM)
    - Region is created by hardware (memory controller)
  - You need to reboot to enable the created region
- Make namespace
  - Namespace is similar concept against SCSI LUN
  - You can configure it by ndctl command
- Format Filesystem (storage or Filesystem DAX)
  - If you would like to use Filesystem DAX, you need to select ext4 or xfs which support Filesystem DAX
    - Currently, if you select xfs, reflink option must be disabled
- Mount Filesystem (storage or Filesystem DAX) with –o dax option
- Make pool (Filesystem DAX or Device DAX) by the PMDK tool if necessary

Users of NVDIMM at kernel layer

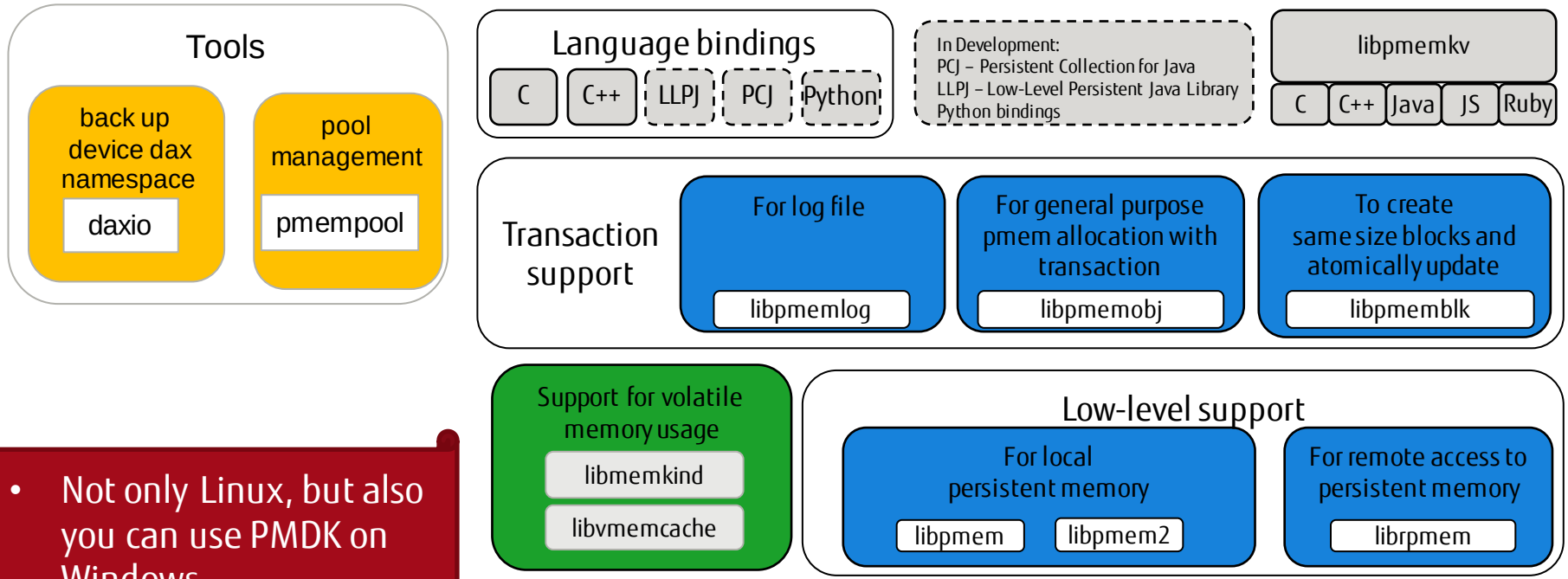
FUJITSU

- dm-writecache
  - Persistent cache for write at device mapper layer
    - dm-cache can use SSD as cache, dm-writecache can use NVDIMM
    - This presentation is helpful
      - https://github.com/ChinaLinuxKernel/CLK2019/blob/master/clk2019-dm-writecache-04.pdf
- Kernel shows Device dax as DRAM
  - Though Intel DCPMM has “Memory Mode” which user can use it as huge RAM, it has some pains
    - Size of DRAM disappears, because it becomes just cache of NVDIMM
    - Software cannot choose area between DRAM or DCPMM
  - In this kernel feature, system RAM size becomes sums of DRAM and NVDIMM
    - Use App Direct Mode and Device DAX namespace
    - Use Memory Hot-add Device DAX area
    - Becomes NUMA node

PMDK (1/2)

FUJITSU

- Set of libraries and tools for Filesystem DAX and Device DAX



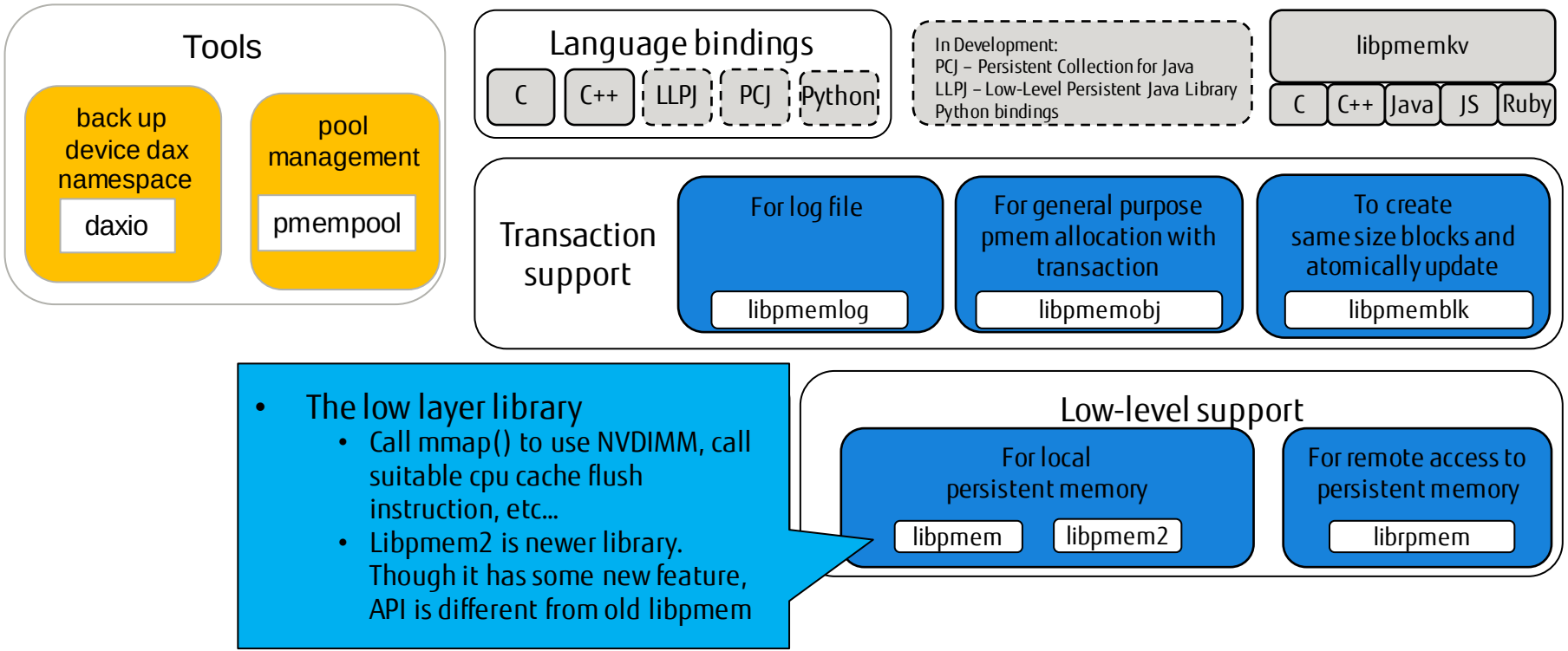
- Not only Linux, but also you can use PMDK on Windows

Note
- Not all components are included in this figure
- To be precise libmemkind is not member of PMDK, but it is recommended library

PMDK (2/2)

FUJITSU

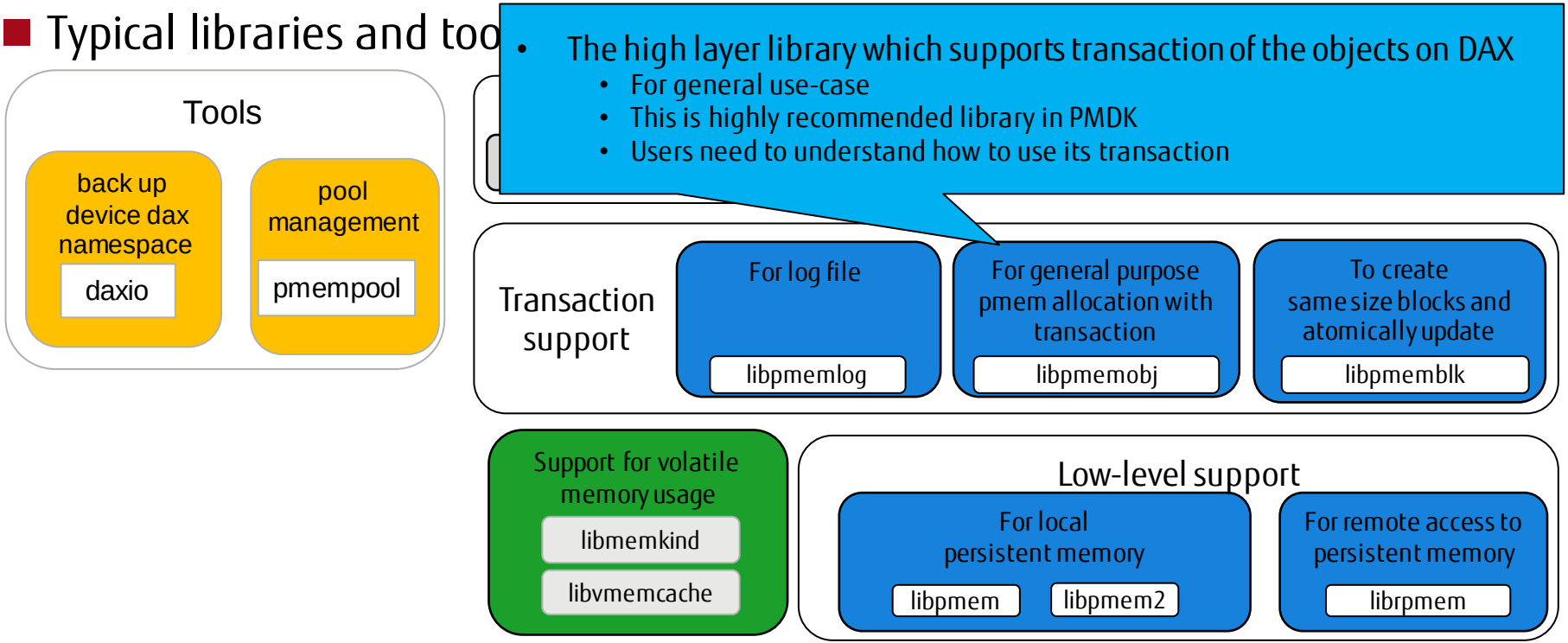
Typical libraries and tools



PMDK (2/2)

FUJITSU

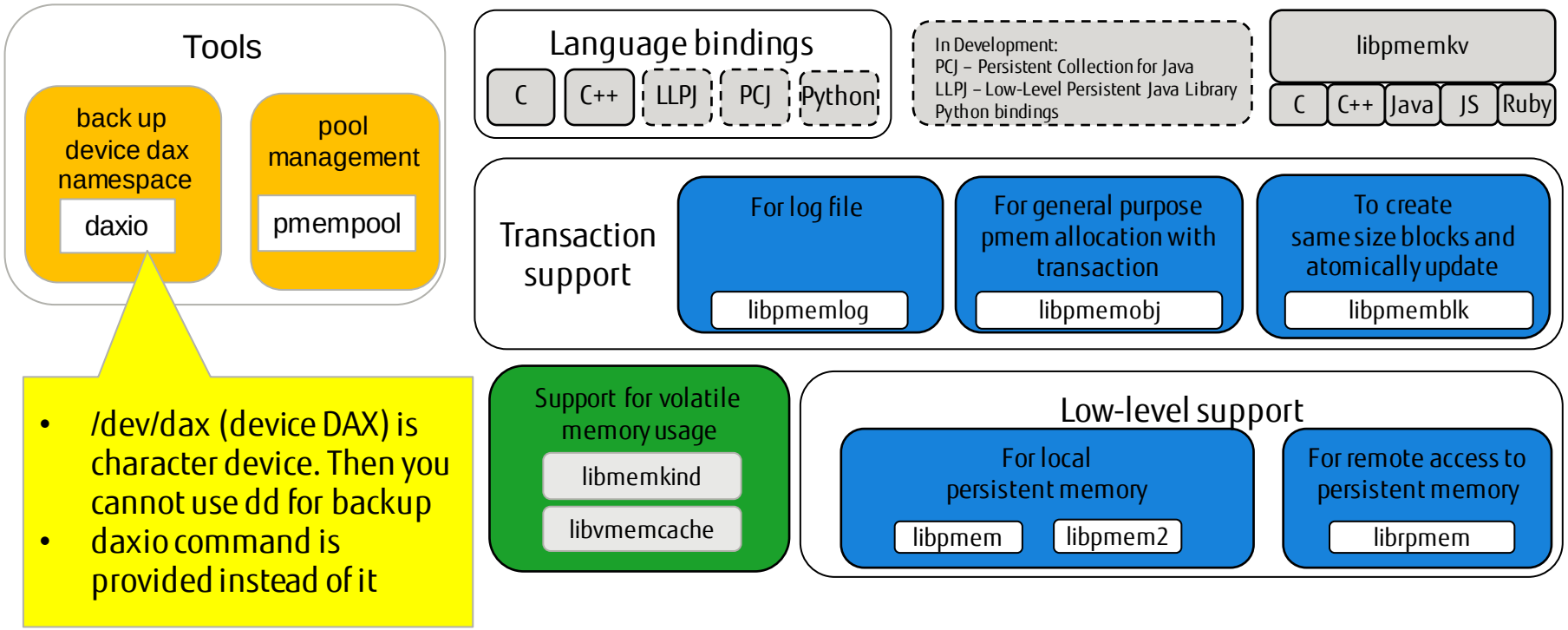
- Typical libraries and too



PMDK (2/2)

FUJITSU

Typical libraries and tools



What is new of libpmem2

FUJITSU

- New low-layer library
  - Introduce new concept “GRANULARITY”
    - PMEM2_GRANULARITY_PAGE : for traditional SSD/HDD
    - PMEM2_GRANULARITY_CACHELINE : for persistent memory (the case for process needs flush cache to make persistency)
    - PMEM2_GRANULARITY_BYTE : for persistent memory (the case for platform support cpu cache persistency)
  - Introduce new functions to get unsafe shutdown status and bad block
    - This library uses library of ndctl command internally to get these information
- Its interface is different from old libpmem

Library for RDMA (1/2)

FUJITSU

- I think RDMA is becoming important for NVDIMM
  - DAX offers direct access method for local NVDIMM
    - However, modern system is a set of many computers which is connected by network
    - So, remote access is also important
  - Traditional network stacks is too heavy to access remote NVDIMM

RDMA is a good way to skip some redundant processing to access remote NVDIMM

  - Use case
    - For scalability
      - Ex) Distributed filesystem, Key Value Store, etc....
    - For make data replication
      - Replace of NVDIMM module is difficult (as I talked at China Linux Kernel Developer Conference)
        https://www.slideshare.net/ygotokernel/the-ideal-and-reality-of-nvdimm-ras-newer-version
    - Etc.

Library for RDMA (2/2)

FUJITSU

- librpma
  - The 2nd library for RDMA
    - The 1st library is librpmem of PMDK, but it is experimental
      - Not popular with users
    - Librpma has been developed with user's requirement
  - Characteristics
    - Relatively easier interface than libibverbs
    - Consideration of making persistency for remote NVDIMM
      - Though hardware can return ack before write completion, user can confirm it with librpma
      - See: https://www.openfabrics.org/wp-content/uploads/2020-workshop-presentations/202.-gromadzki-ofa-workshop-2020.pdf
        (*) The above table is quoted from this presentation

PAIN POINTS OF THE FIRST APPROACH

Customer's feedback to librpmem

PMDK (librpmem) provides	Customers expect
Replication process **tightly coupled to libpmemobj**	Replication process **controlled by app**
Poolsets semantic is used as replication basis	Replication process to **follow app data semantic**
Static replication **configuration**	Replication **configuration** might **change online** based on application needs
No access to replicated data in runtime	At least **read access** to replicated data in runtime
Focus on RDMA.Write API	RDMA.Write/Send as well as RDMA.Read support depending on application case
	Neither libfabric nor SSH dependencies

6

OpenFabrics Alliance Workshop 2020