China Linux Kernel Developer Conference 2019
Oct 19, 2019 14:30~15:10

FUJITSU
shaping tomorrow with you

Critical steps to remove the experimental of Filesystem-DAX

QI Fuli / Ruan Shiyang

Who are we

FUJITSU

- QI Fuli
  - Software Engineer at Fujitsu Ltd
  - PhD Student at University of Tokyo
  - Working on Persistent Memory
  - Email: qi.fuli@fujitsu.com
- Ruan Shiyang
  - Software Engineer at Fujitsu Ltd
  - Working on Persistent Memory
  - Email: ruansy.fnst@cn.fujitsu.com

Contents

FUJITSU

- Introduction of NVDIMM
- Critical steps
- Summary

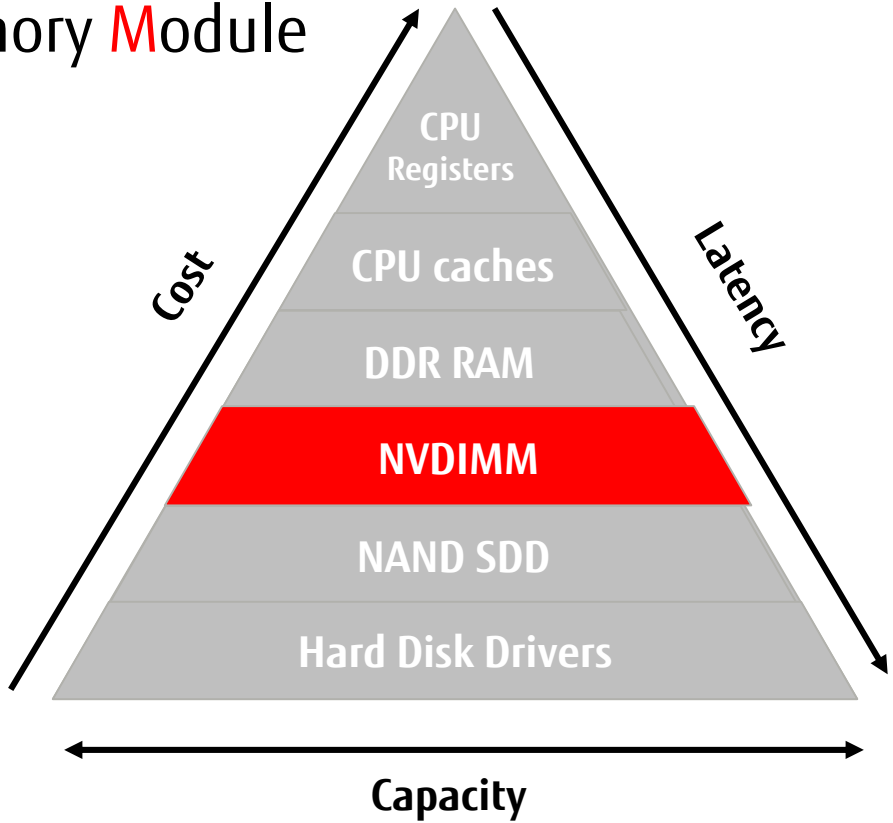
FUJITSU

Introduction of NVDIMM

NVDIMM Overview

FUJITSU

- Non-Volatile Dual In-line Memory Module
  - a type of random-access memory
  - NVDIMM retains its data even if electrical power is removed
- Use case
  - In-Memory Database, etc.



Concepts [1]

FUJITSU

- Interleave set
  - Two or more NVDIMMs create an N-Way interleave set to provide stripes read/write operations for increased throughput
- Namespace
  - Defines a contiguously-addressed range of Non-Volatile Memory
- Region
  - A group of one or more NVDIMMs, or an interleaved set, that can be divided up into one or more Namespaces

[1] https://docs.pmem.io/ndctl-users-guide/concepts

Concepts [1]

FUJITSU

Type

- Defines the way in which the persistent memory associated with a Namespace or Region can be accessed
- PMEM: Direct access to the media via load/store operations. (DAX supported)
- BLK: Direct access to the media via Apertures. (DAX is not supported)

Mode

- Defines which NVDIMM software feature are enabled for a given Namespace.
- Namespace Modes include fsdax, devdax, sector, and raw.

[1] https://docs.pmem.io/ndctl-users-guide/concepts

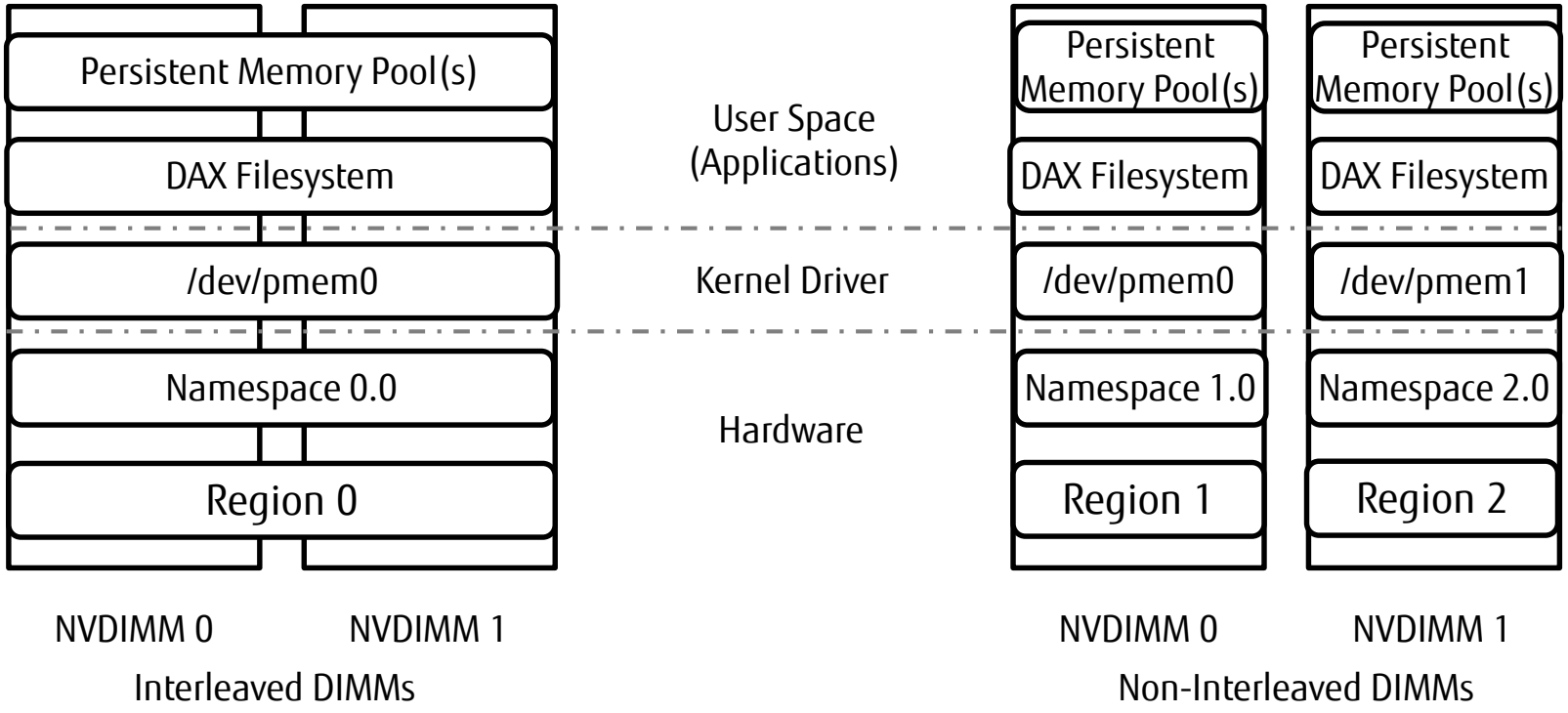
DAX

FUJITSU

- Filesystem-DAX
  - creates a block device(/dev/pmemX[.Y])
  - removes the page cache from the I/O path
  - allows mmap() to establish direct mappings to persistent memory media
- Device-DAX
  - intended for applications that mmap() the entire capacity
  - creates a character device (/dev/daxX.Y) instead of a block device

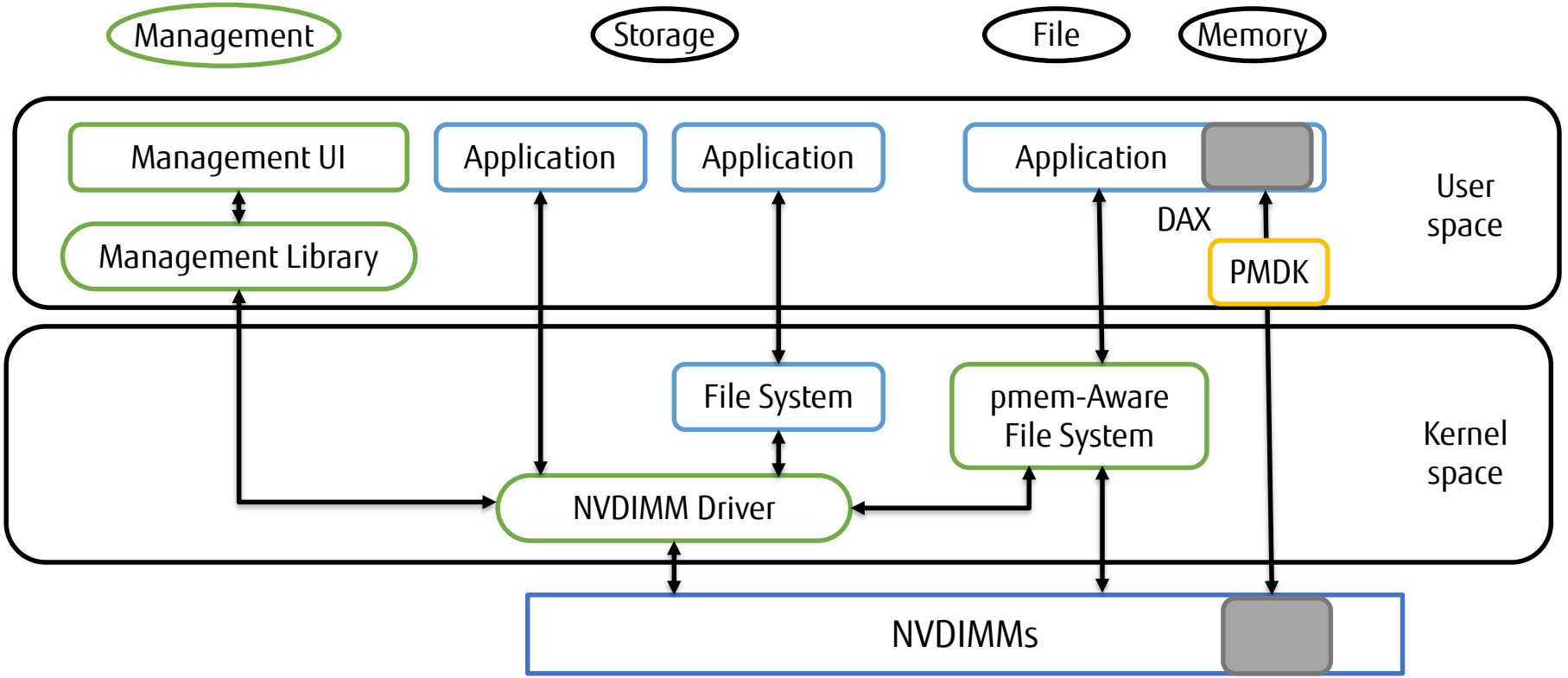
Configuration options

FUJITSU



NVM Programming Model

FUJITSU



Non-Volatile Device Control (NDCTL)

FUJITSU

- A utility for managing the Linux LIBNVDIMM Kernel subsystem
- Working with various NVDIMMs from different vendors
- Operations supported by ndctl
  - Provisioning capacity
  - Enumerating Devices
  - Enabling and Disabling NVDIMMs, Regions, and Namespaces
  - Managing NVDIMM Labels

Sample of using filesystem-dax

FUJITSU

```
# ndctl create-namespace -e "namespace0.0" -m fsdax –f
{"dev":"namespace0.0",
  "mode":"fsdax",
  "map":"dev",
  "size":"7.87 GiB (8.45 GB)",
  "uuid":"0b10e1bb-b6ae-4600-bec3-4bc40f7b8f07",
  "sector_size":512,
  "align":2097152,
  "blockdev":"pmem0"}
# ls /dev | grep pmem
pmem0
```

-m fsdax,
define the namespace mode fsdax

Sample of using filesystem-dax

FUJITSU

```
# parted /dev/pmem0
GNU Parted 3.2
Using /dev/pmem0
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel gpt
(parted) mkpart
Partition name?  []? nvdimm
File system type?  [ext2]? xfs
Start? 1M
End? 8G
# ls /dev | grep pmem
pmem0
pmem0p1
```

Sample of using filesystem-dax

FUJITSU

```
# sudo mkfs.xfs /dev/pmem0p1
meta-data=/dev/pmem0p1           isize=512    agcount=4, agsize=515840 blks
         =                       sectsz=4096  attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=0
data     =                       bsize=4096   blocks=2063360, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=4096  sunit=1 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
# mkdir /mnt/fsdax
# mount -o dax /dev/pmem0p1 /mnt/fsdax
```

-o dax, /mnt/fsdax/ can be directly accssed

FUJITSU

Critical steps

Critical steps

FUJITSU

- Index
  - Support reflink for fsdax
  - Memory map for fsdax
  - The "dax" semantics
- Start from XFS
  - A widely used filesystem, used as default filesystem in RHEL and CentOS.
  - Ext4 doesn't support reflink.
  - Btrfs is in progress.

FUJITSU

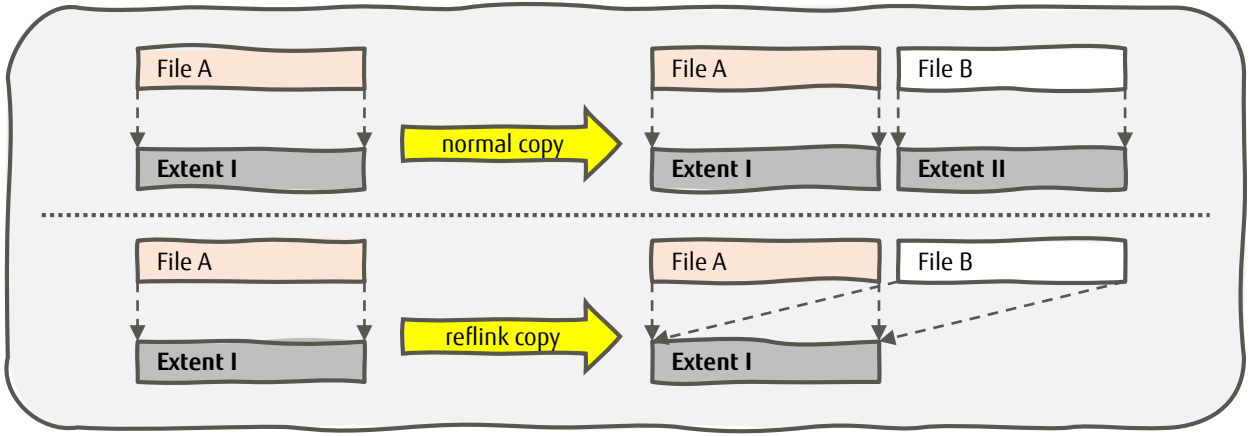
Critical steps

- Support reflink for fsdax
- Memory unmap for fsdax
- The "dax" semantics

Reflink supported

FUJITSU

- What is reflink?
  - Files share extents for same data



  - Advantages

    Fast copy

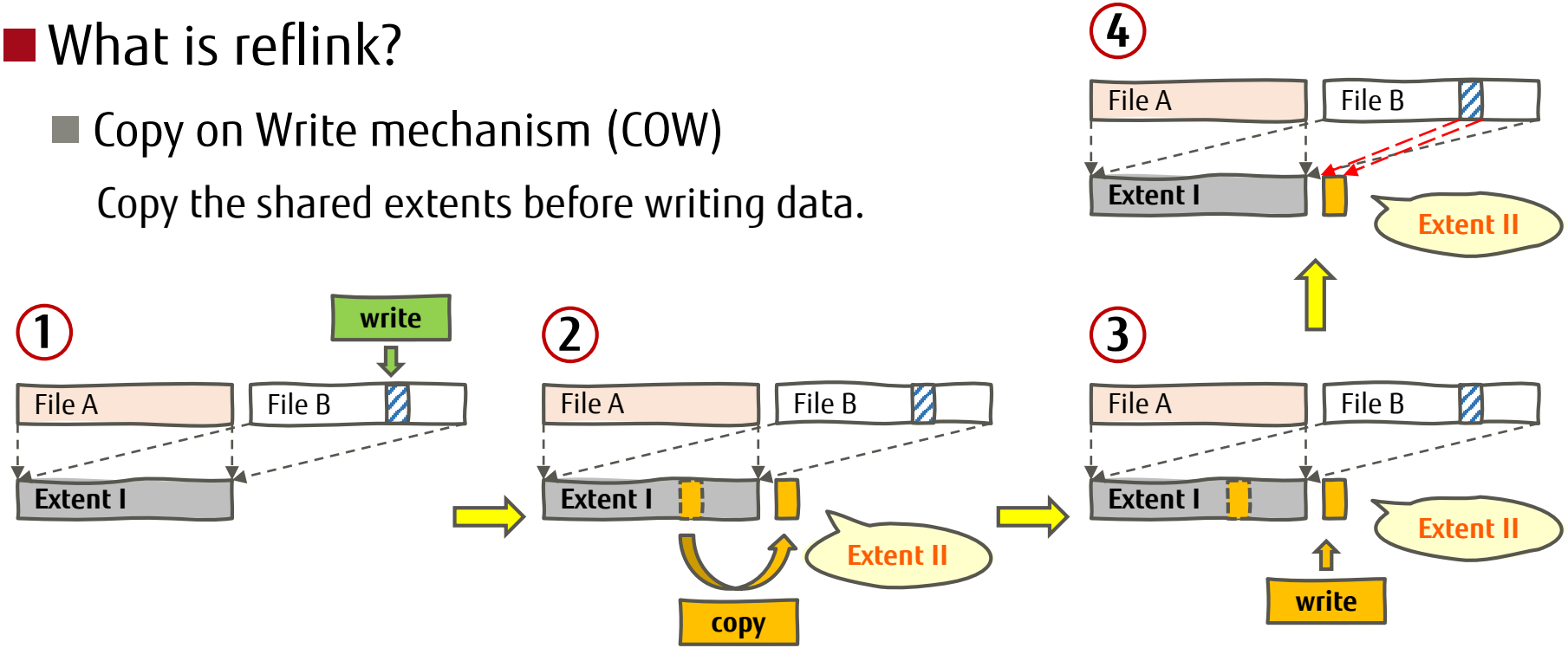
    Save storage

Reflink supported

FUJITSU

- What is reflink?
  - Copy on Write mechanism (COW)

    Copy the shared extents before writing data.

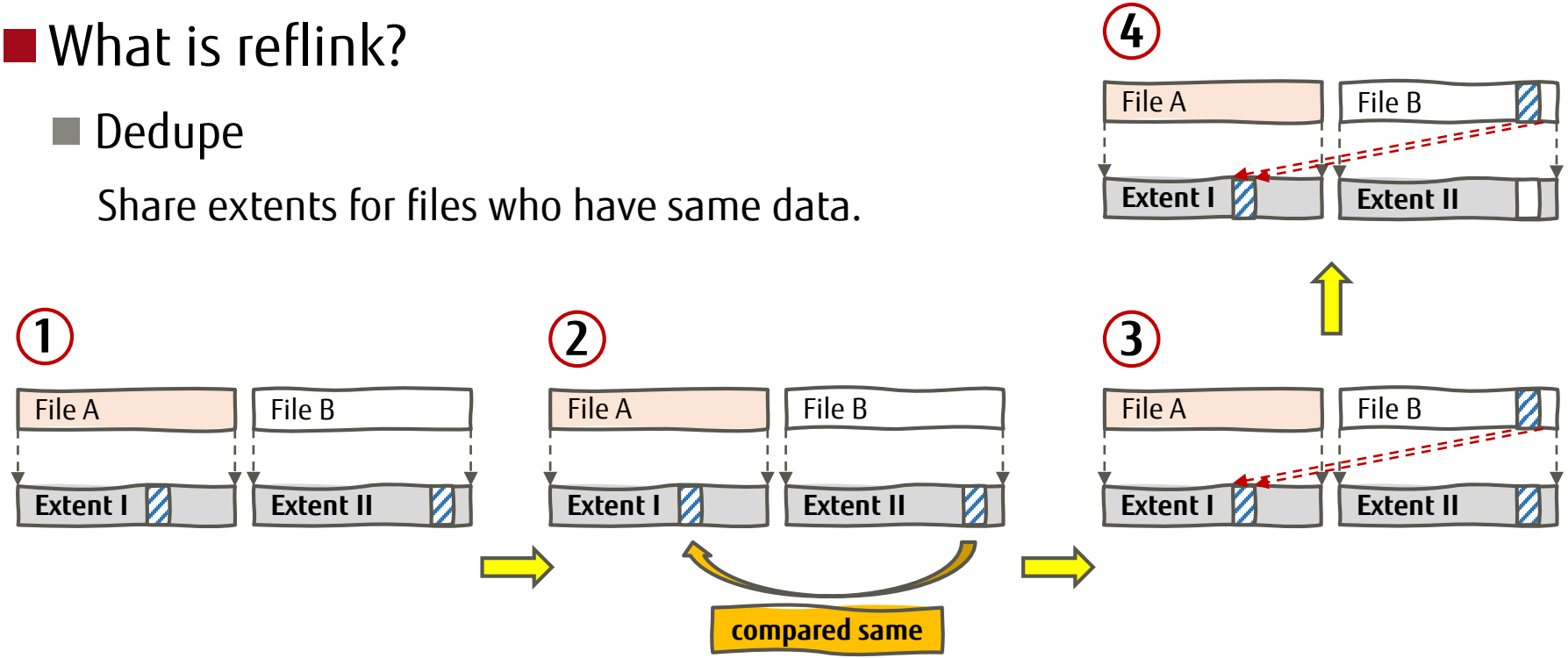


Reflink supported

FUJITSU

- What is reflink?
  - Dedupe

    Share extents for files who have same data.



Reflink supported

FUJITSU

- How to enable reflink?
  - Add '**-m reflink=1**' when making a filesystem

```
$ mkfs.xfs -m reflink=1 /path/to/device
```

  - Use reflink feature when copying

```
$ cp --reflink=always fileA fileB
```

Reflink supported

FUJITSU

- Time cost

```
$ time cp file1G.bin file1G.1.bin
real        0m4.498s
user        0m0.014s
sys         0m3.942s

$ time cp --reflink=always file1G.bin file1G.2.bin
real        0m0.008s
user        0m0.001s
sys         0m0.006s
```

Very fast

- Disk usage

```
$ ll -h
total 2.0G
-rw-rw-r--. 1 ryan ryan 1.0G Oct 18 19:26 file1G.1.bin
-rw-rw-r--. 1 ryan ryan 1.0G Oct 18 19:26 file1G.2.bin
-rw-rw-r--. 1 ryan ryan 1.0G Oct 18 19:20 file1G.bin
$ df -h /mnt
Filesystem      Size  Used Avail Use% Mounted on
/dev/pmem0      4.0G  2.1G  1.9G  53% /mnt
```

Not occupied

Fsdax supported.

FUJITSU

- What is fsdax?
  - A mode of a NVDIMM namespace

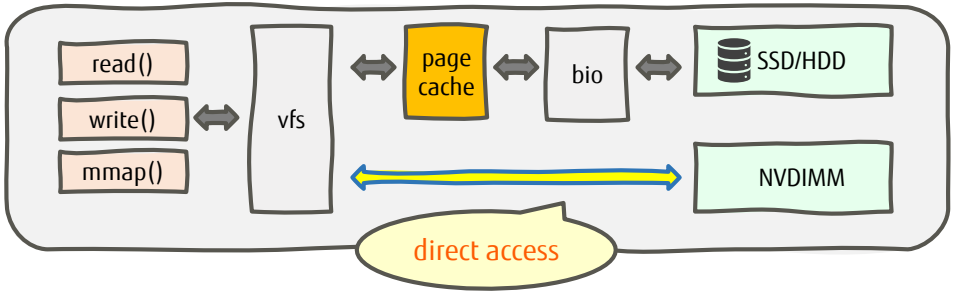
    Create a filesystem on pmem and access data through VFS.

    No need to change apps' code.
  - Bypass page cache

    Copy data directly between pmem device and apps.

    No block io.

    No page cache.



Fsdax supported.

FUJITSU

- How to enable fsdax?
  - add '**-o dax**' when mounting a pmem device

```
$ mount -o dax /path/to/pmem /path/to/mountpoint
```

  - Enables DAX flag for all files. *

* Will talk in section: The "dax" semantics.

Didn't support both reflink & fsdax

FUJITSU

- Try to enable them together
  - make a reflink featured XFS and mount it with dax option

```
$ mkfs.xfs -m reflink=1 [...] && mount -o dax [...]
```

  - then error occurs

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

  - dmesg shows

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

Didn't support both reflink & fsdax

FUJITSU

- Reason
  - There are some restriction code in XFS to avoid enabling these two feature together since they are unfinished for now.

    Unexpected error will happen, and it may damage your data. It's dangerous.

Force enable them

FUJITSU

- What will happen?
  - The '**copy --reflink=always**' command works.

```
$ cp --reflink=always fileA fileB
```

fileA and fileB do share same extents.

```
$ xfs_io -c "fiemap" /mnt/*
fileA:
        0: [0..55]: 160..215
fileB:
        0: [0..55]: 160..215
```

physical bno

logic bno

Force enable them

FUJITSU

- What will happen?
  - When writing data to one of these files, no one changed.

    New extent did be allocated.

    Metadata did not be updated.

    COW not work correctly.

```
$ xfs_io -c “pwrite -S 0xAB 4k 1k” fileA

$ xfs_io -c "fiemap" /mnt/*
fileA:
        0: [0..55]: 160..215
fileB:
        0: [0..55]: 160..215
```

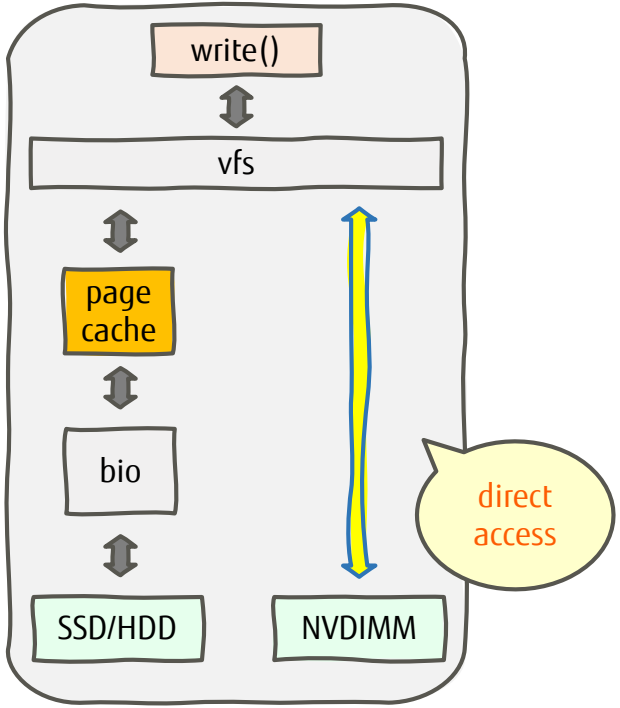
physical bno

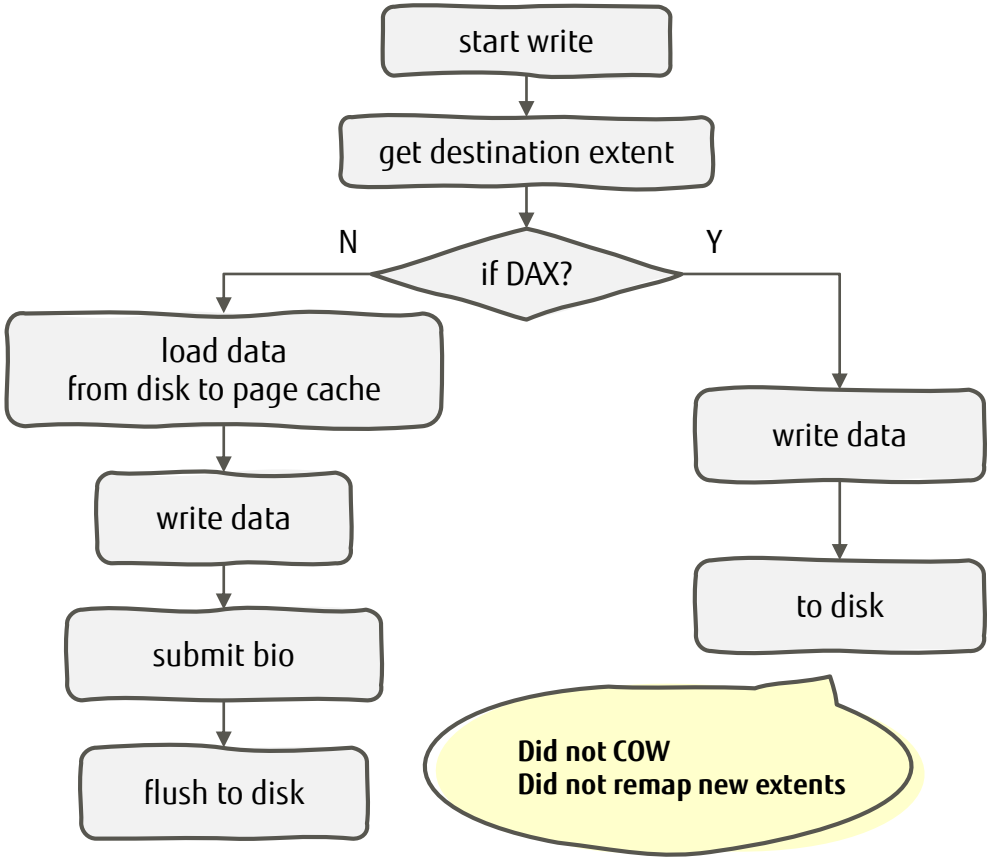
logic bno

Force enable them

FUJITSU

- Why that happened?





Iomap model

FUJITSU

- XFS uses **iomap model** to handle write operation.

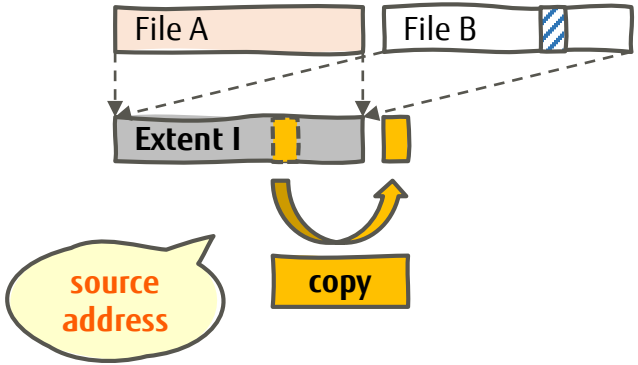
```
-> iomap_apply()      /* start write operation */
  -> iomap_begin()    /* get the disk offset where write at */
  -> actor()          /* perform the write operation */
  -> iomap_end()      /* commit and/or unreserve space previous allocated */
```

Handler functions	
`xfs_file_iomap_begin()`	add **ALLOCATE** new extents for **COW**
`dax_iomap_actor()`	add **COPY** source data to new allocated extents, and then, **WRITE** new data
`xfs_file_iomap_end()`	add **UPDATE** the extent list of this file

Add a “srcmap”

FUJITSU

- The source address for COW
  - COW operation executed in **->actor()** needs to know where to copy from.
  - Get source address in **->iomap_begin()**.



Add a “srcmap”

FUJITSU

- How?
  - At first, we added a field ‘**src_addr**’ in struct iomap to remember the source address. And It worked.
  - After discussion, community decided to add another iomap called ‘**srcmap**’ to do this job.
  - And add a new type called ‘**IOMAP_COW**’ for **->actor()** to distinguish COW operation with others.

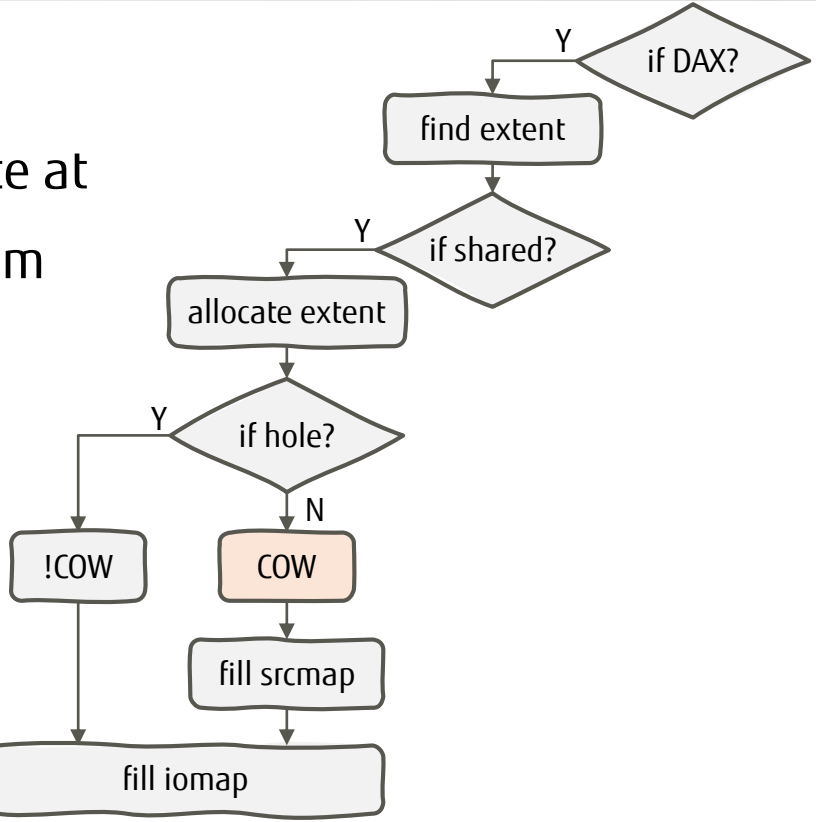
```
+ #define IOMAP_COW 0x06   /* copy data from srcmap before writing */

  int (*iomap_begin)(struct inode *inode, loff_t pos, loff_t length,
-                    unsigned flags, struct iomap *iomap);
+                    unsigned flags, struct iomap *iomap,
+                    struct iomap *srcmap);
```

Fill “srcmap”

FUJITSU

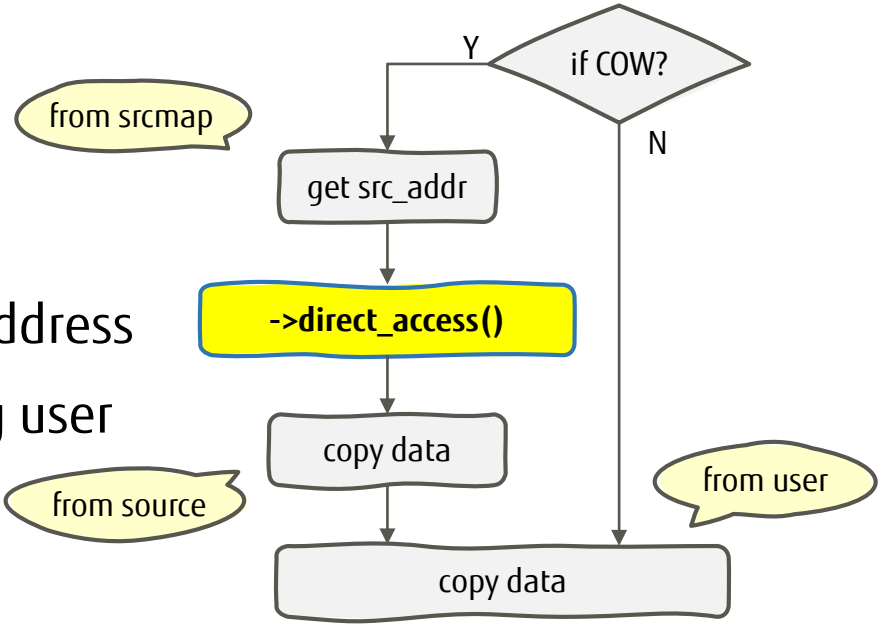
- Fill “**srcmap**”
  - iomap: the destination extent to write at
  - srcmap: the source extent to copy from
  - Filled in **->iomap_begin()**
- How?
  - Add handle for file who has dax flag and shared extents.



Add COW for write()

FUJITSU

- Perform COW in **write()** path
  - The dax driver provides **->direct_access()** to get physical memory address in pmem
  - Data is being written in **->actor()**
- How?
  - Copy source data safely from source address to destination address before copying user data to destination address.



Add COW for mmap()

FUJITSU

- Perform COW in **mmap()** path
  - Access to the virtual memory address that **mmap()** gave calls page fault, which is handled by **dax_iomap_pte_fault()**, or by **dax_iomap_pmd_fault()** in case of huge pages.

Normal page	Huge page
`dax_iomap_pte_fault()`	`dax_iomap_pmd_fault()`

  - This also uses iomap model, but data is not being written here. Just allocate the virtual memory address.
- How?
  - Familiar with **write()** path, copy data before virtual address is associated.

After COW

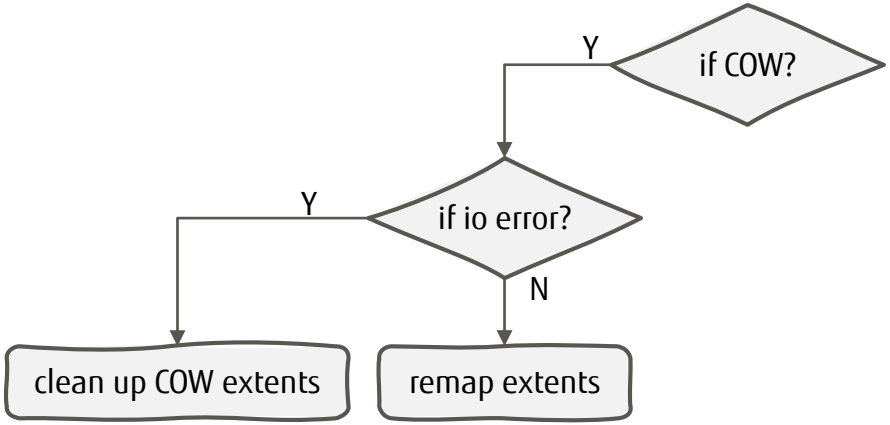
FUJITSU

- Update extent list

  Since new extent allocated, the file need to remap it.

- How?

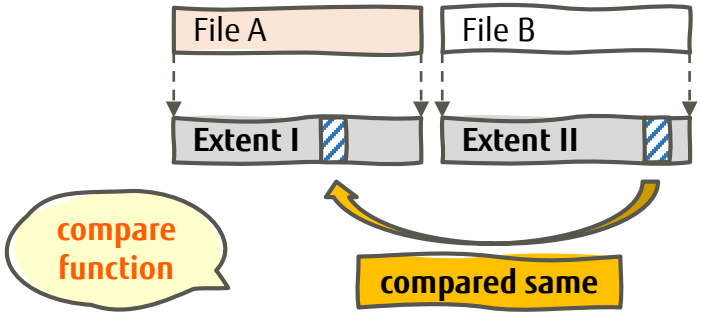
  Execute **xfs_reflink_end_cow()** in **->iomap_end()** if it is a COW operation.



Add a “dax” dedupe

FUJITSU

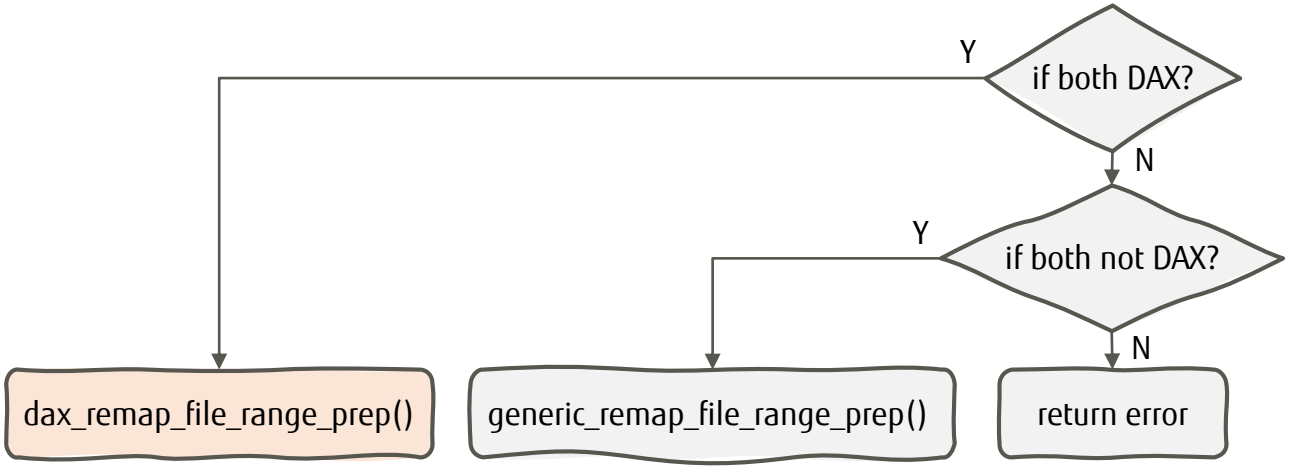
- Handle dedupe
  - Dedupe uses **generic_remap_file_range_prep()** to compare two extents byte-by-byte to tell if they are same.
  - However, that function is for general usage. It compares extents cached in memory(page cache). Not suitable to fsdax.



Add a “dax” dedupe

FUJITSU

- How?
  - Add a fsdax specific compare function and call it if files both have DAX flag.
  - Don’t share extents between a DAX file with a non-DAX file.



Support reflink for fsdax

FUJITSU

- Features of XFS
  - Reflink supported
  - Fsdax supported
  - But didn't support both reflink & fsdax yet
- What to do to support them together?
  - Iomap model
  - Add a "srcmap"
  - Fill "srcmap"
  - Add COW for write()
  - Add COW for mmap()
  - After COW
  - Add a "dax" dedupe

FUJITSU

Critical steps

- Support reflink for fsdax
- Memory unmap for fsdax
- The "dax" semantics

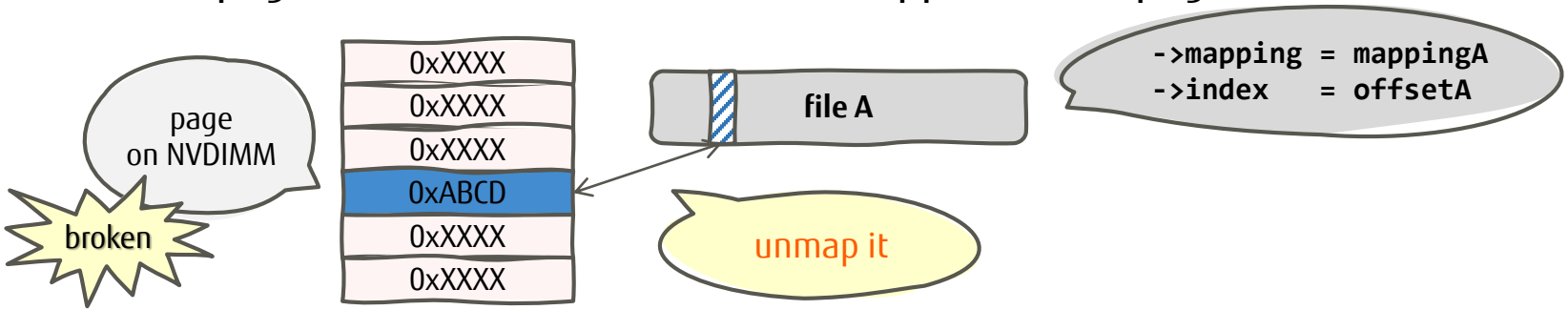
Memory unmap for fsdax

FUJITSU

- Munmap
  - Appears in pair with **mmap()**.

    In general case, page remembers which file belongs to by **->mapping**, and which offset locates in by **->index**.
  - Also called when Memory Failure.

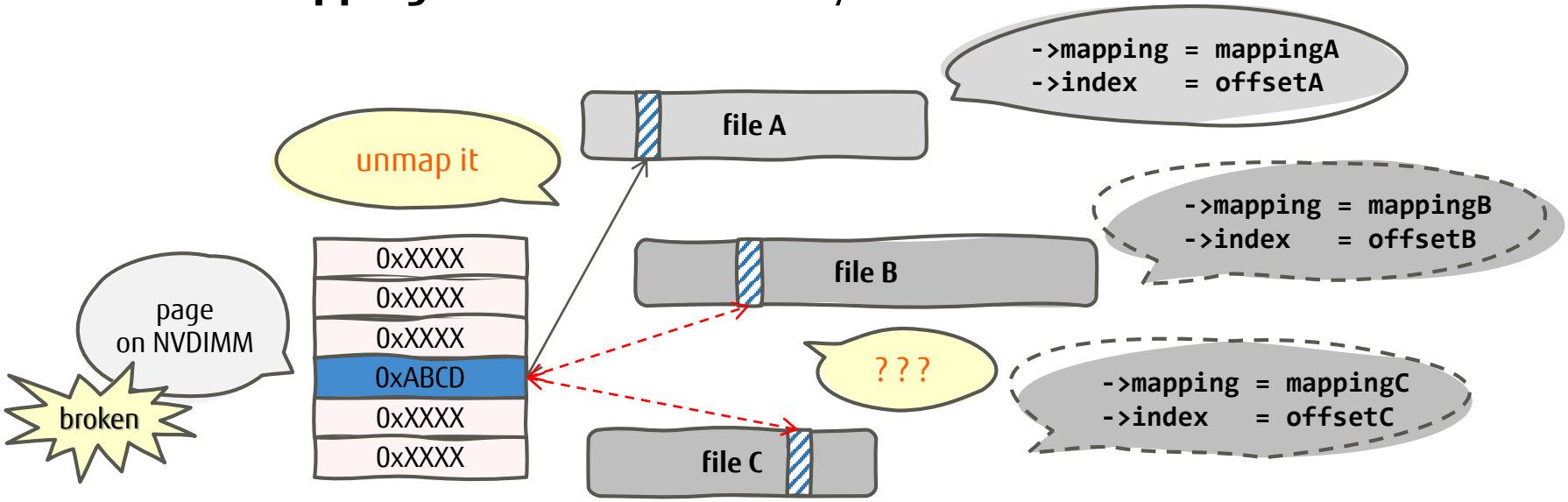
    When a page broken, files whose extent mapped to this page need to be unmapped.



Memory unmap for fsdax

FUJITSU

- Reflink case
  - One page on NVDIMM may belongs to multi files.
  - But the **->mapping** and **->index** can only save for one file.



FUJITSU

Critical steps

- Support reflink for fsdax
- Memory unmap for fsdax
- The "dax" semantics

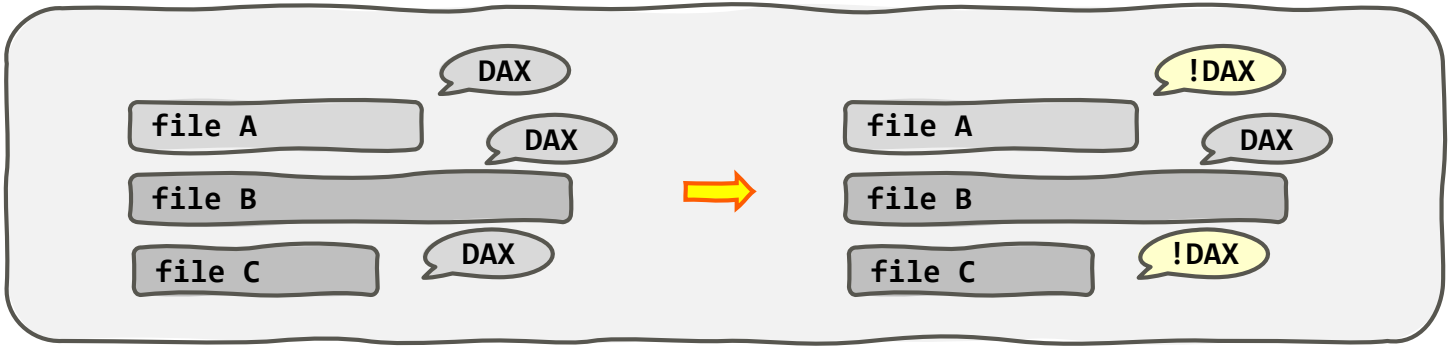
The "dax" semantics

FUJITSU

- After mount with option **'-o dax'**
  - ALL files have dax flag set.
  - Users only want to set dax flag on some specific files.

    Write operation on NVDIMM is a bit slower than on RAM.

    In another word, fsdax write may slower than buffered write in some case.



The "dax" semantics

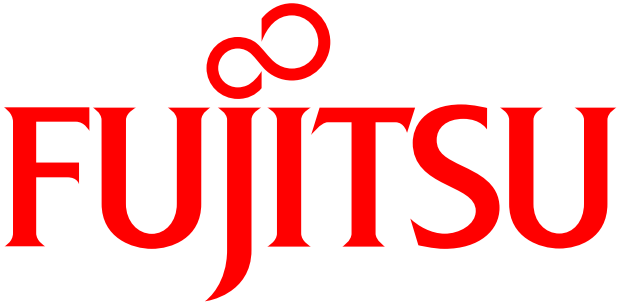
FUJITSU

- Still under discussion...
- Remove “-o dax”?
  - Auto enable DAX flag on dax capable device. Drop **“-o dax**”.
- How to enable the functionality at a finer granularity than a mount option?
  - Change file’s attribute to determine if enable dax or not.
- How to set DAX flag?
  - Initial set when a new file created.
  - Change the flag of files already created.

Summary

FUJITSU

- What is NVDIMM
- How to use it
- How to support reflink for fsdax
- The Memory unmap problem
- The “dax” semantic problem



shaping tomorrow with you