STORAGE INDUSTRY SUMMIT

Convergence of Storage and Memory Developing the Needed Ecosystem

JANUARY 20, 2016, SAN JOSE, CA

Jeff Moyer
Red Hat
Principal Software Engineer

Persistent Memory in Linux

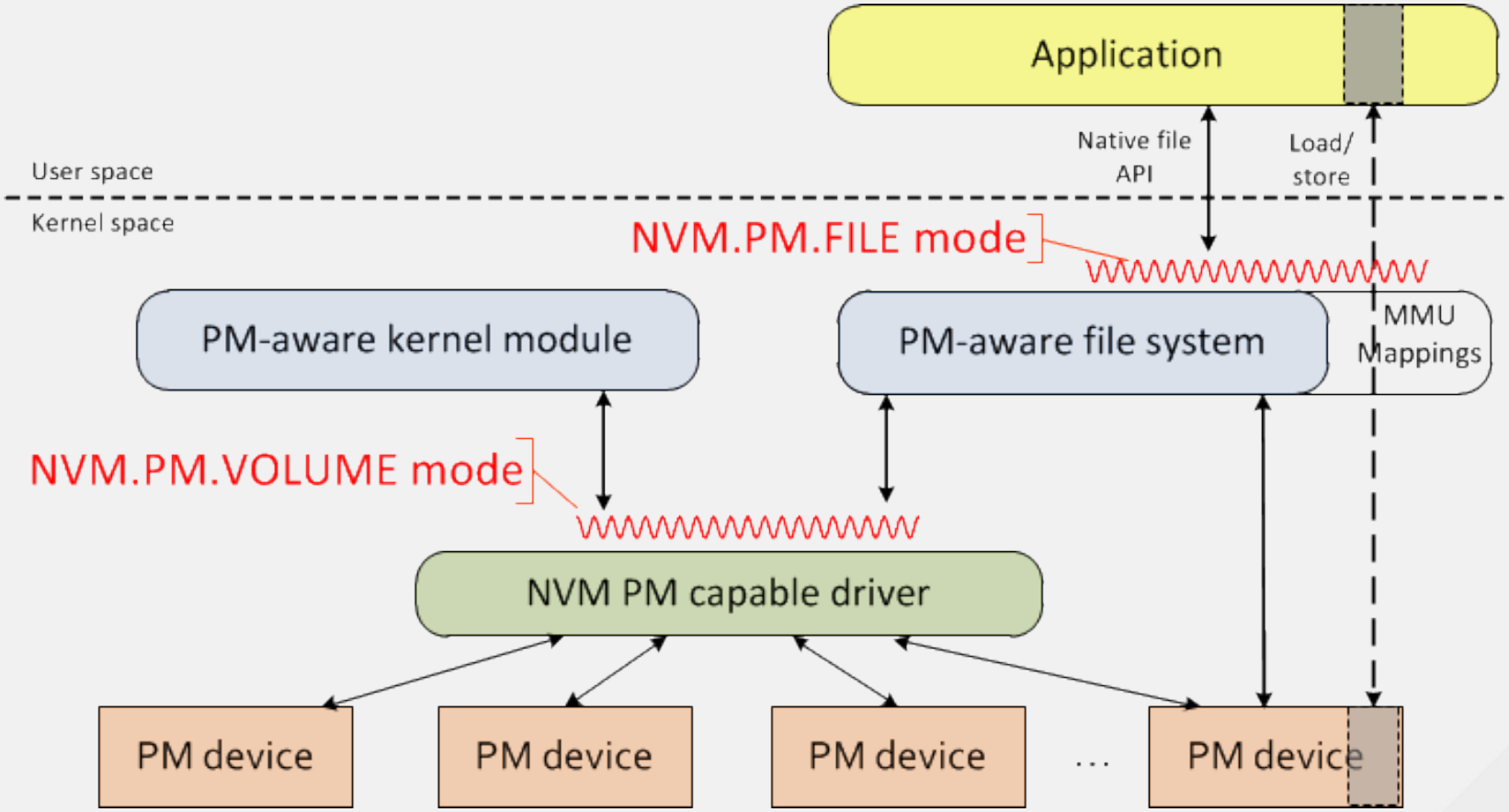
SNIA SSSI | SOLID STATE STORAGE

AGENDA

- Software Architecture
- Status
- Getting Started with PMEM

SOFTWARE ARCHITECTURE

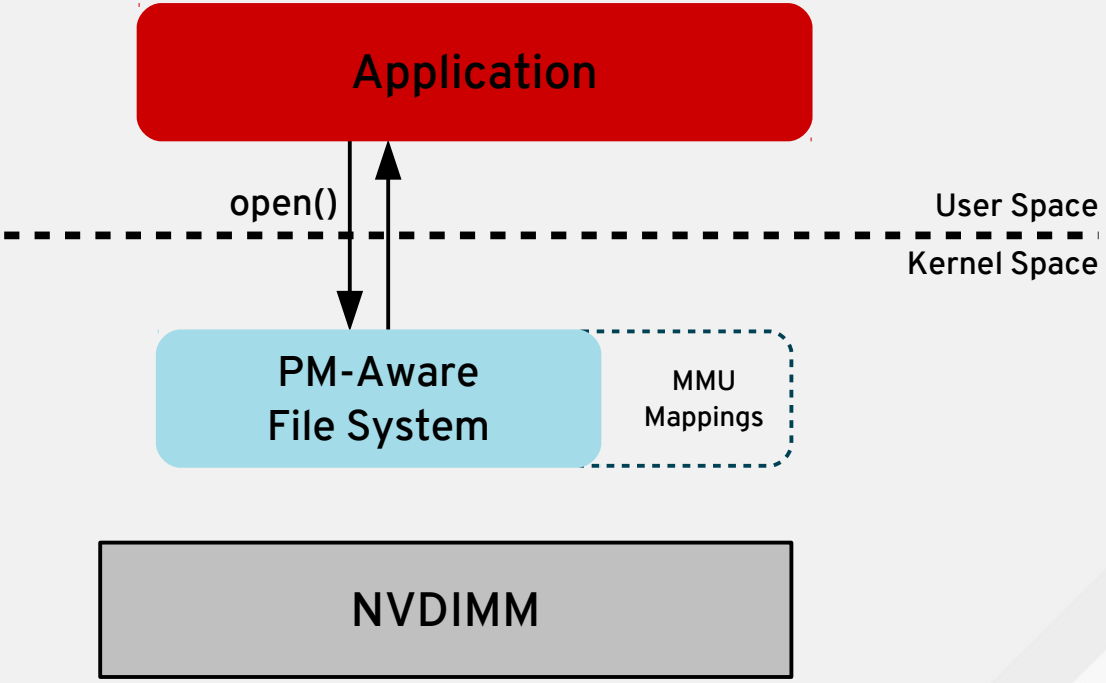
NVM.PM MODES



source: NVM Programming Model Specification

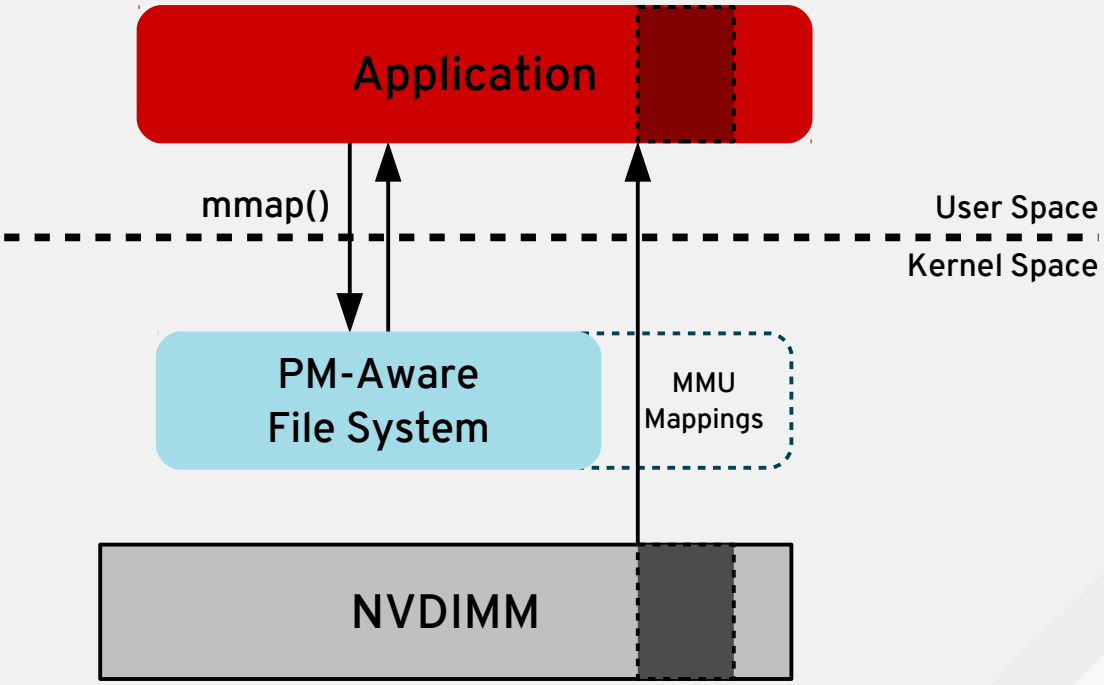
APPLICATION VIEW

- Step 1: open()



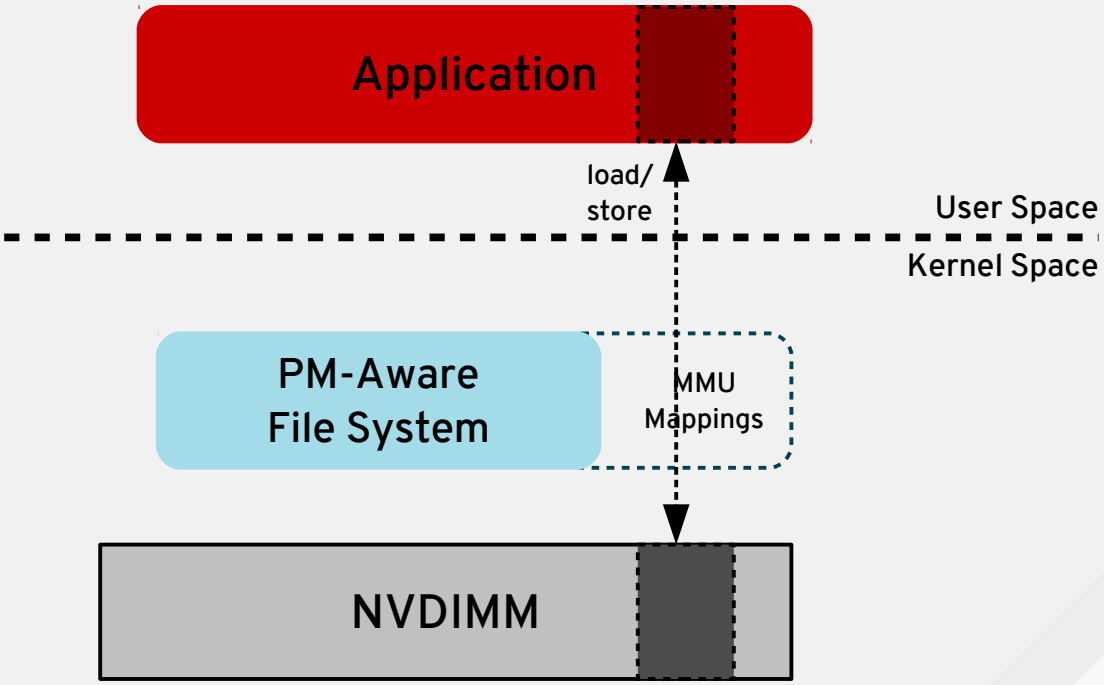
APPLICATION VIEW

- Step 1: open()
- Step 2: mmap()



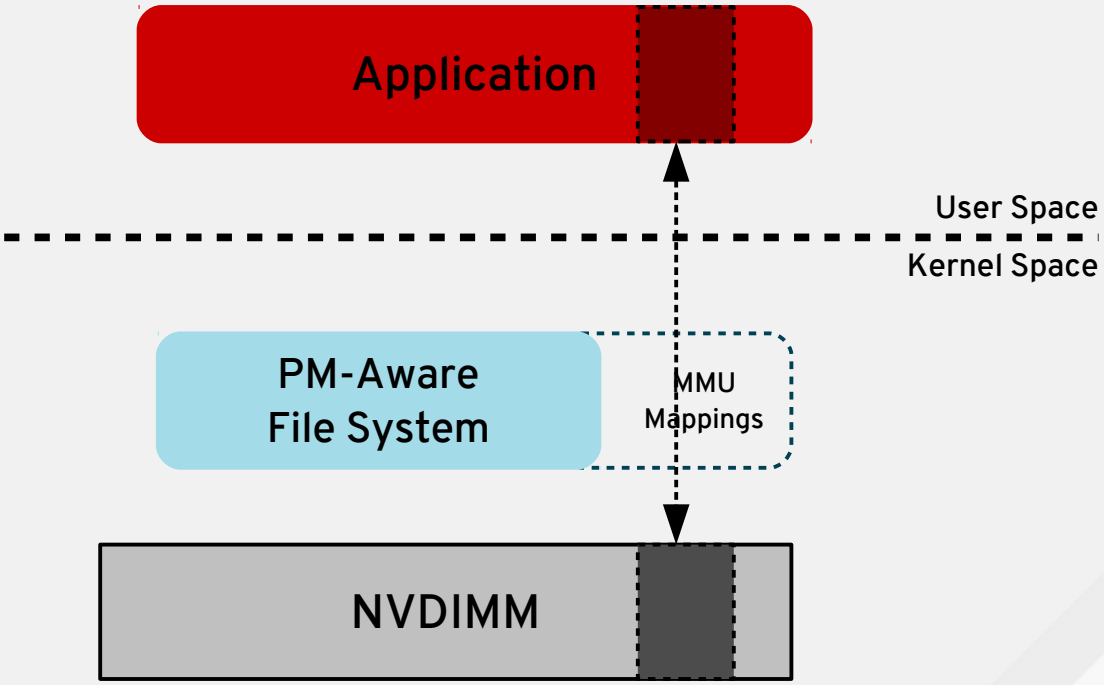
APPLICATION VIEW

- Step 1: open()
- Step 2: mmap()
- Step 3: ???

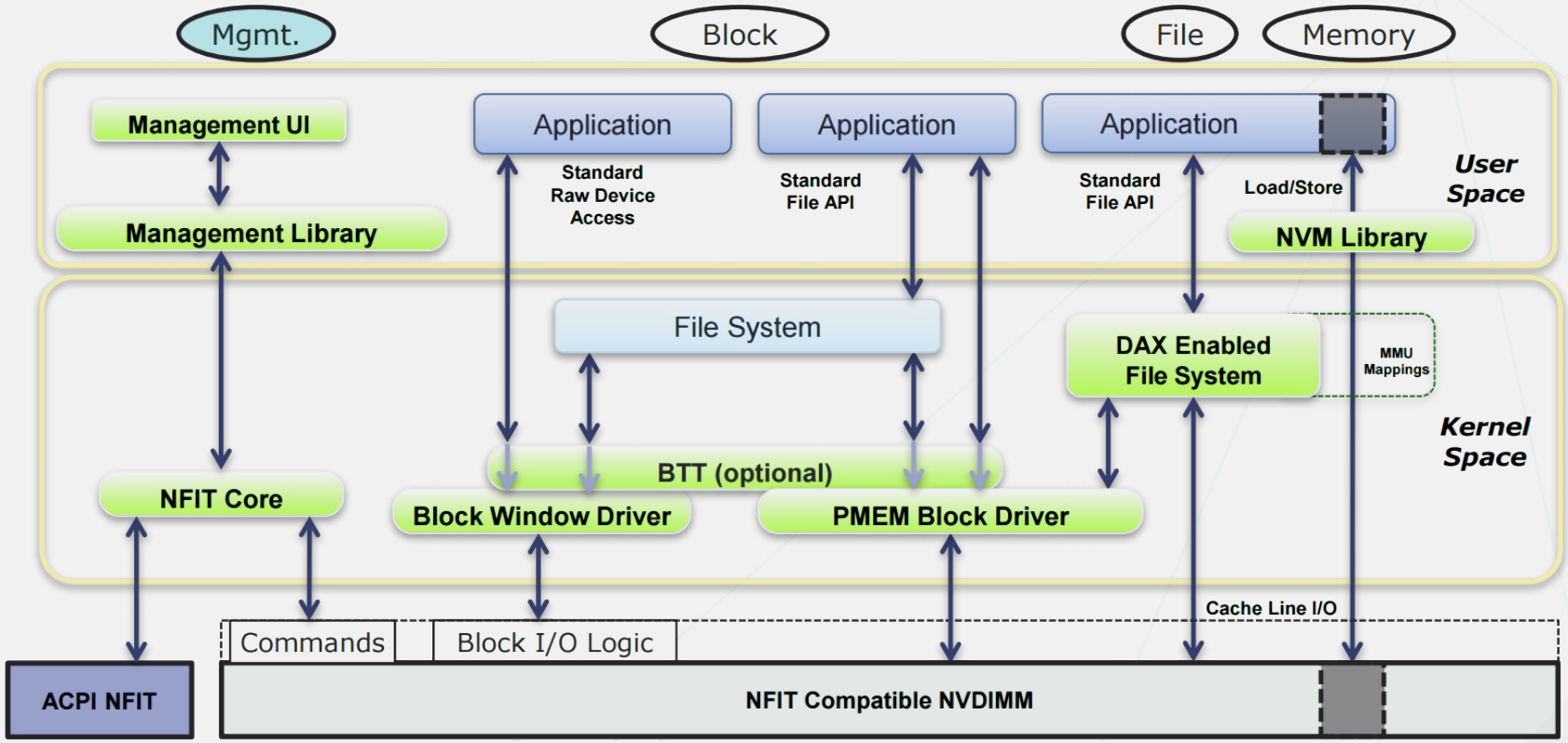


APPLICATION VIEW

- Step 1: open()
- Step 2: mmap()
- Step 3: ???
- Step 4: **PROFIT!!!**



NVDIMM SOFTWARE ARCHITECTURE



source: http://pmem.io/documents/NVDIMM_Namespace_Spec.pdf

NVM.PM.FILE ACTIONS

- Implemented by kernel
  - Map
  - Sync
- Implemented in userspace
  - Optimized_flush
  - Optimized_flush_and_verify

POSIX READ/WRITE WITH DAX

- Bypasses the Page Cache
- Synchronous

POSIX READ/WRITE WITH DAX

- Bypasses the Page Cache
- Synchronous

Still suffers from torn sectors!

ERROR HANDLING

- 2 cases to handle
  - Machine checks
  - Known bad “blocks”

MACHINE CHECK EXCEPTIONS

- Goal: Turn MCEs into SIGBUS
- Not recoverable on all platforms (or in all cases)

KNOWN BAD BLOCKS

- Address Range Scrub
- Stored per block device

STATUS

SUBSYSTEMS ADDED/MODIFIED

IN SUPPORT OF NVDIMMS

- ACPI
- Device Drivers
- Architecture code (e.g. x86)
- Block Layer
- VFS
- File Systems (ext4, xfs)
- Memory Management

NON-KERNEL COMPONENTS

- QEMU, libvirt, virt-manager
- Libraries (nvml, libndctl)
- Kexec
- Anaconda (installer)
- Smartmontools
- Management utilities

WHAT WORKS?

Kernel

- NVDIMM driver core
  - NFIT parsing, pmem driver, nd_blk, btt
- DAX support (experimental)
  - 4k + 2MB pages
  - Ext4, xfs
- Bad block tracking **infrastructure**
- Core dump filtering

WHAT IS MISSING?

Kernel

- DAX
  - fsync/msync
  - 1GB page support in file systems
  - Error handling
- Management utilities

GETTING STARTED

TRYING OUT PMEM

- Kernel configurations
- Making and mounting a file system
- Testing the programming model

TRYING OUT PMEM

Kernel Configuration

```
CONFIG_ZONE_DEVICE=y
CONFIG_X86_PMEM_LEGACY_DEVICE=y
CONFIG_X86_PMEM_LEGACY=m
CONFIG_ACPI_NFIT=m
CONFIG_LIBNVDIMM=m
CONFIG_BLK_DEV_PMEM=m
CONFIG_ND_BLK=m
CONFIG_ND_CLAIM=y
CONFIG_ND_BTT=m
CONFIG_BTT=y
CONFIG_ND_PFN=m
CONFIG_NVDIMM_PFN=y
```

… or just use Fedora

TRYING OUT PMEM

Device Nodes

Kernel Parameter Added

- memmap=16G!16G

```
# ls /dev/pmem*
/dev/pmem0

# fdisk -l /dev/pmem0

Disk /dev/pmem0: 16 GiB, 17179869184 bytes,
33554432 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

TRYING OUT PMEM

Making and mounting a file system

```
# mkfs -t xfs -d su=1g,sw=1 /dev/pmem0
# mount -t xfs -o dax /dev/pmem0
```

OR

```
# mkfs -t ext4 /dev/pmem0
# mount -t ext4 -o dax /dev/pmem0
```

NOTE: Inconsistent behavior

- ext4 fails if DAX unavailable
- xfs logs a message

Using DAX

```
int
main(int argc, char **argv)
{
     int         fd;
     void       *addr;
     struct stat st;

     fd = open(argv[1], O_RDWR);
     fstat(fd, &st);
     addr = mmap(0, st.st_size, PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);

...
```

Using DAX

Verifying PMEM Was Allocated

```
/proc/self/smaps:

7fc173c6b000-7fc17a06b000 rw-s 00000000 103:00
2097155                   /mnt/nvm/testfile
Size:             102400 kB
Rss:                   0 kB
Pss:                   0 kB
Shared_Clean:          0 kB
Shared_Dirty:          0 kB
Private_Clean:         0 kB
Private_Dirty:         0 kB
Referenced:            0 kB
Anonymous:             0 kB
AnonHugePages:         0 kB
Shared_Hugetlb:        0 kB
Private_Hugetlb:       0 kB
Swap:                  0 kB
SwapPss:               0 kB
KernelPageSize:        4 kB
MMUPageSize:           4 kB
Locked:                0 kB
VmFlags: rd wr sh mr mw me ms mm hg
```

Using DAX

Verifying PMEM Was Allocated

```
        char *caddr = addr;

        FILE *fp;
        fp = fopen("/proc/self/smaps", "r");

        while (fgets(line, PROCMAXLEN, fp) != NULL) {
                static const char vmflags[] = "VmFlags:";
                static const char mm[] = " mm";

                /* check for range line */
                if (sscanf(line, "%p-%p", &lo, &hi) == 2) {
                        if (needmm) {
                                /* last range matched, but no mm flag found */
                                break;
                        } else if (caddr < lo) {
                                /* never found the range for caddr */
                                break;
                        } else if (caddr < hi) {
                                /* start address is in this range */
...
```

Using DAX

Verifying PMEM Was Allocated

```
        size_t rangelen = hi - caddr;

        /* remember that matching has started */
        needmm = 1;

        /* calculate remaining range to search for */
        if (len > rangelen) {
                len -= rangelen;
                caddr += rangelen;
        } else {
                len = 0;
        }
}
```

Using DAX

Verifying PMEM Was Allocated

```
        } else if (needmm && strncmp(line, vmflags,
                        sizeof (vmflags) - 1) == 0) {
                if (strstr(&line[sizeof (vmflags) - 1], mm) != NULL) {
                        if (len == 0) {
                                /* entire range matched */
                                retval = 1;
                                break;
                        }
                        needmm = 0;     /* saw what was needed */
                } else {
                        /* mm flag not set for some or all of range */
                        break;
                }
        }
}

fclose(fp);
```

… or just use nvml

… or just use nvml

(which is the source of the above stole^wborrowed code)

SUMMARY

- BTT for legacy applications
- PMEM devices for modified applications or applications that are resilient to torn sectors
- NVM.PM.FILE.MAP can be used today with a Fedora kernel
- NVM.PM.FILE.SYNC is coming soon

REFERENCES

- Upstream NVDIMM mailing list:
  - https://lists.01.org/mailman/listinfo/linux-nvdimm
- NVDIMM wiki, includes pointers for getting started:
  - https://nvdimm.wiki.kernel.org/
- pmem.io, includes blog posts, reference materials, and nvml:
  - http://pmem.io
- NVDIMM Namespace Specification (label format, etc):
  - http://pmem.io/documents/NVDIMM_Namespace_Spec.pdf
- SNIA NVM Programming Model Specification:
  - http://www.snia.org/tech_activities/standards/curr_standards/npm

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos