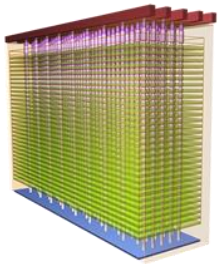
Persistent memory in Linux

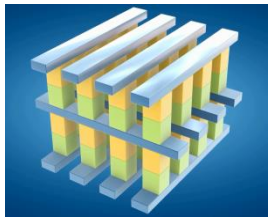
Dmitry Krivenok
EMC

NVM today and tomorrow



STT

Memristors



PCM

ReRAM

PCI EXPRESS

nvm EXPRESS

2015

?

SNIA NVM Programming Model (NPM)

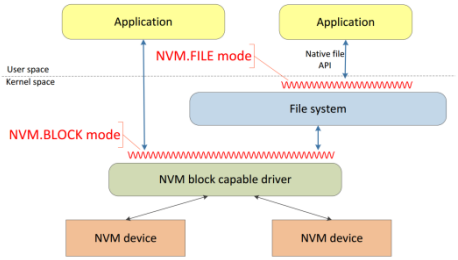
NVM technology agnostic

SNIA

Advancing storage & information technology

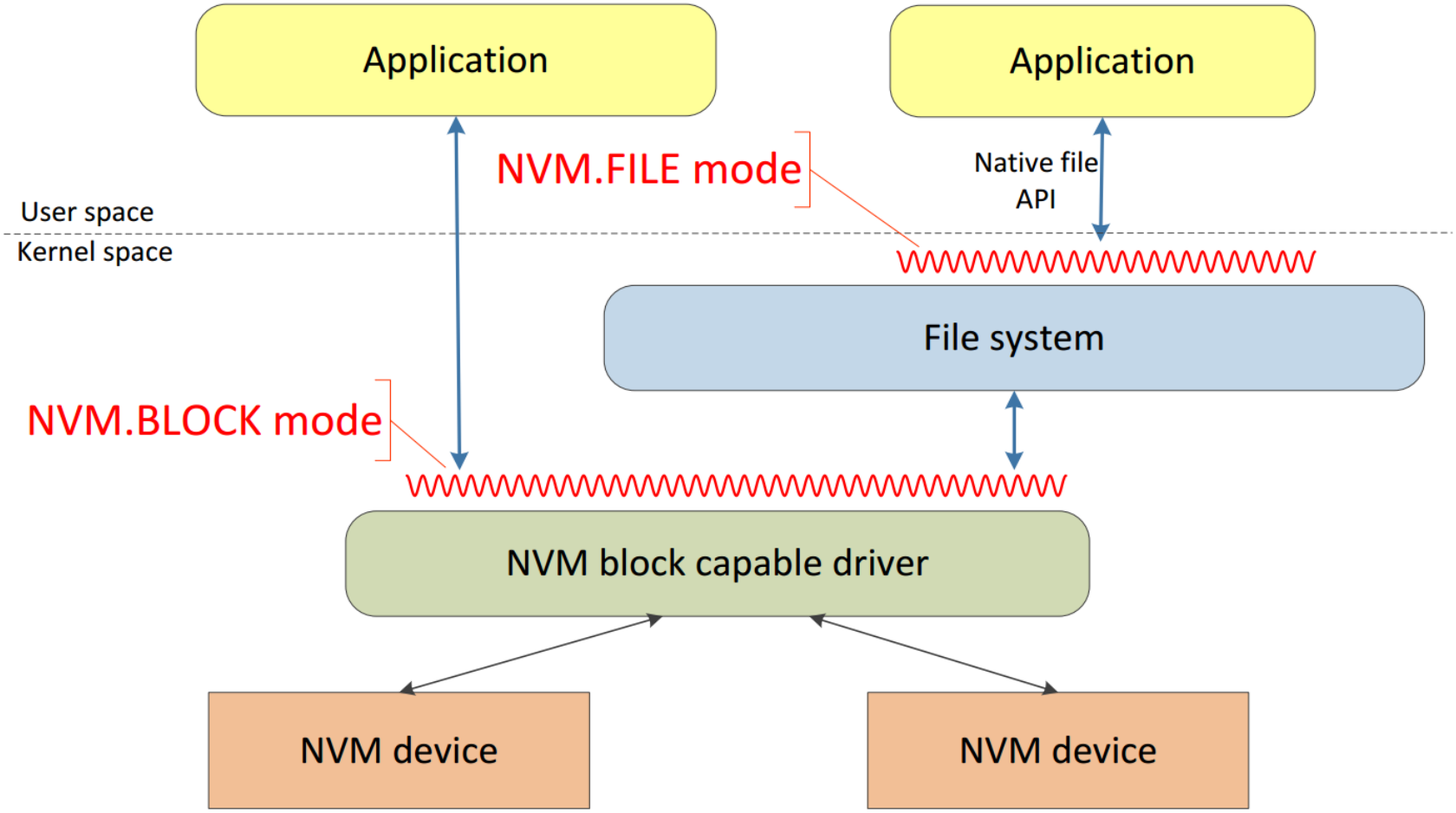
General concepts mappable to OS APIs

4 major programming models



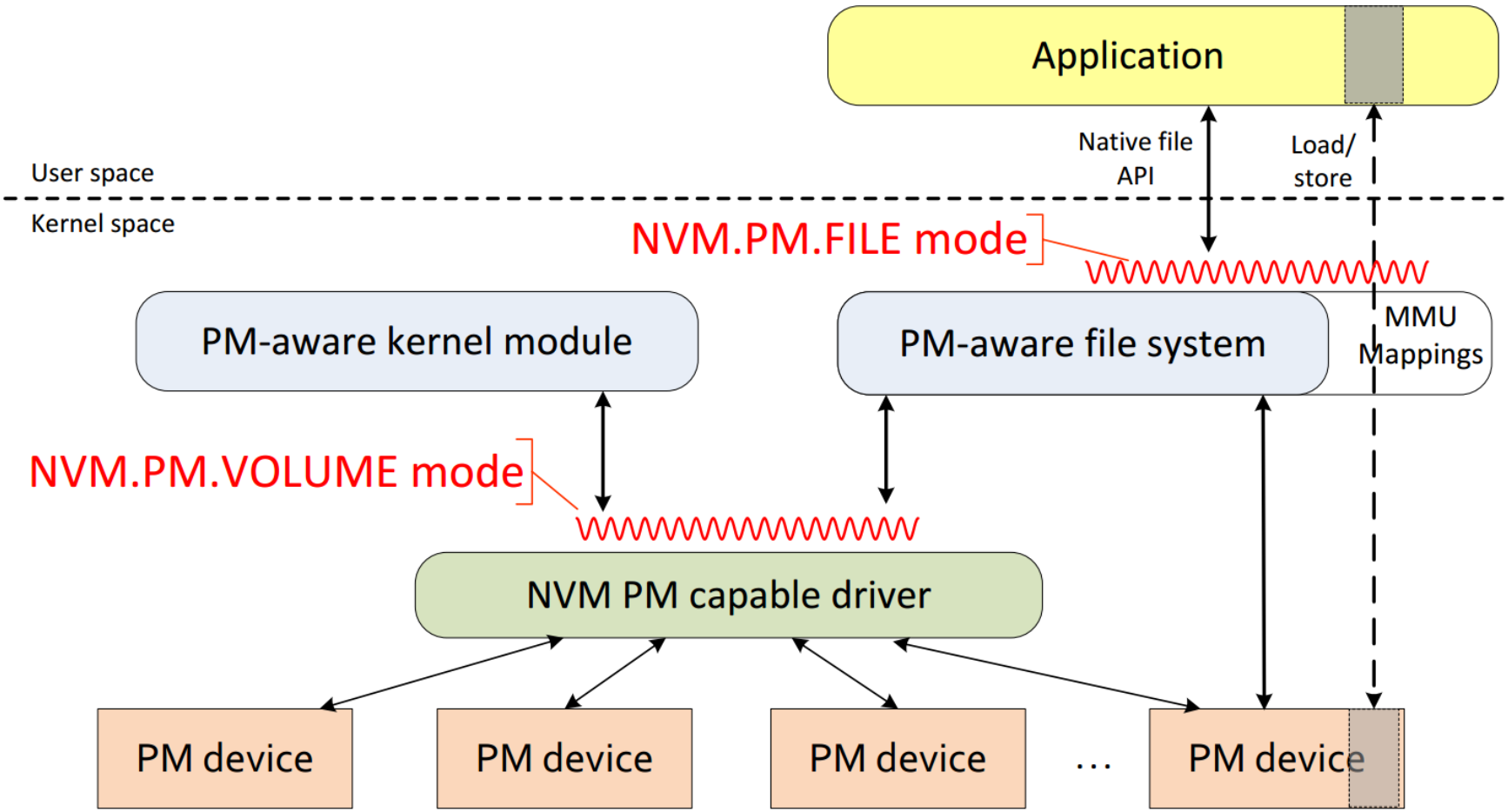
http://www.snia.org/tech_activities/standards/curr_standards/npm

NPM FILE and BLOCK modes



Picture taken from http://www.snia.org/sites/default/files/NVMProgrammingModel_v1.1.pdf

NPM PM.VOLUME and PM.FILE modes



Picture taken from http://www.snia.org/sites/default/files/NVMProgrammingModel_v1.1.pdf

Welcome NVDIMMs

NVDIMMs standardization in 2015

Standard for NVDIMM-N/F

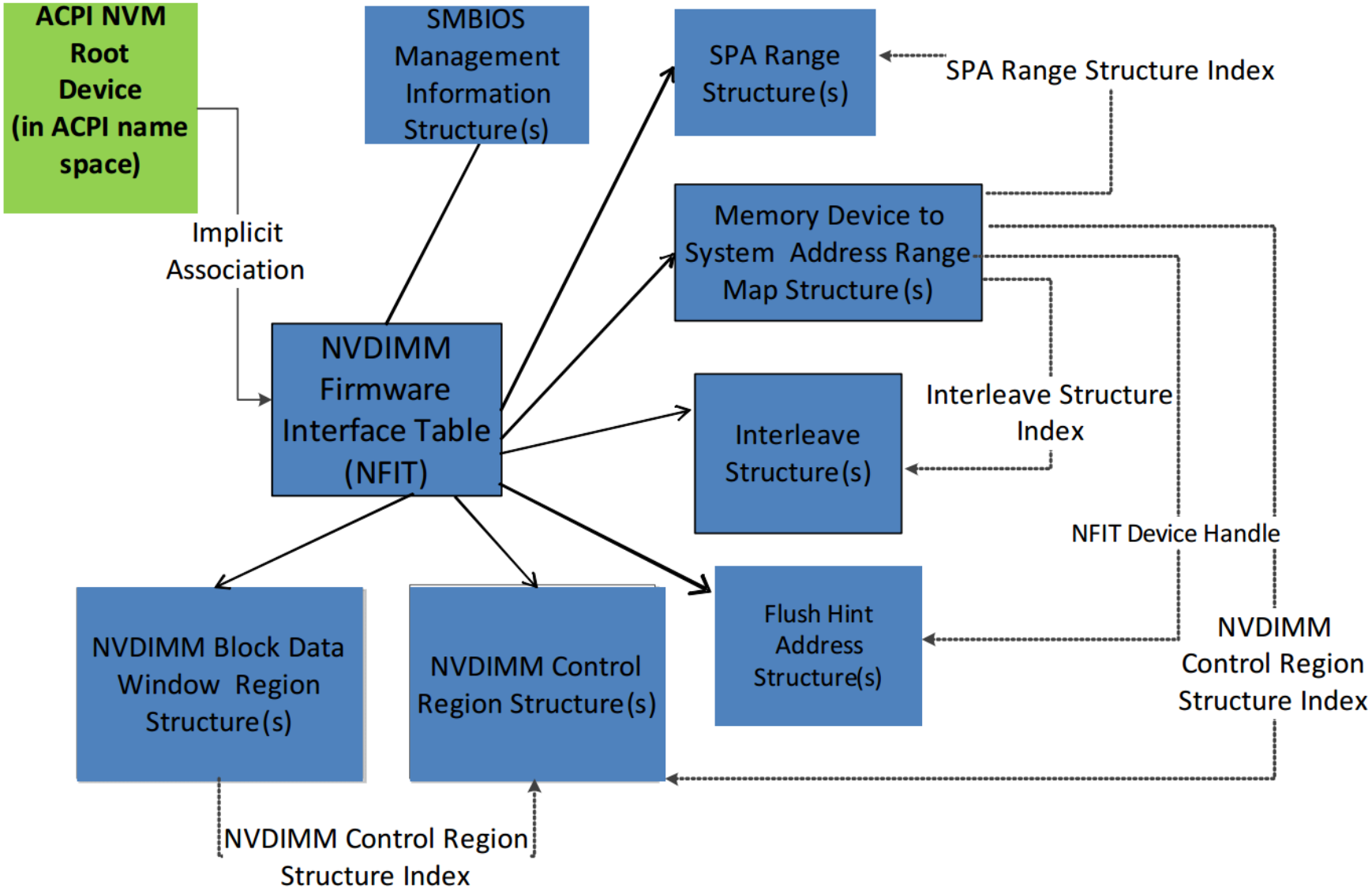
ACPI 6.0 specification

NVDIMM specs (NS, DSM, BW)

Kernel support for NVDIMMs

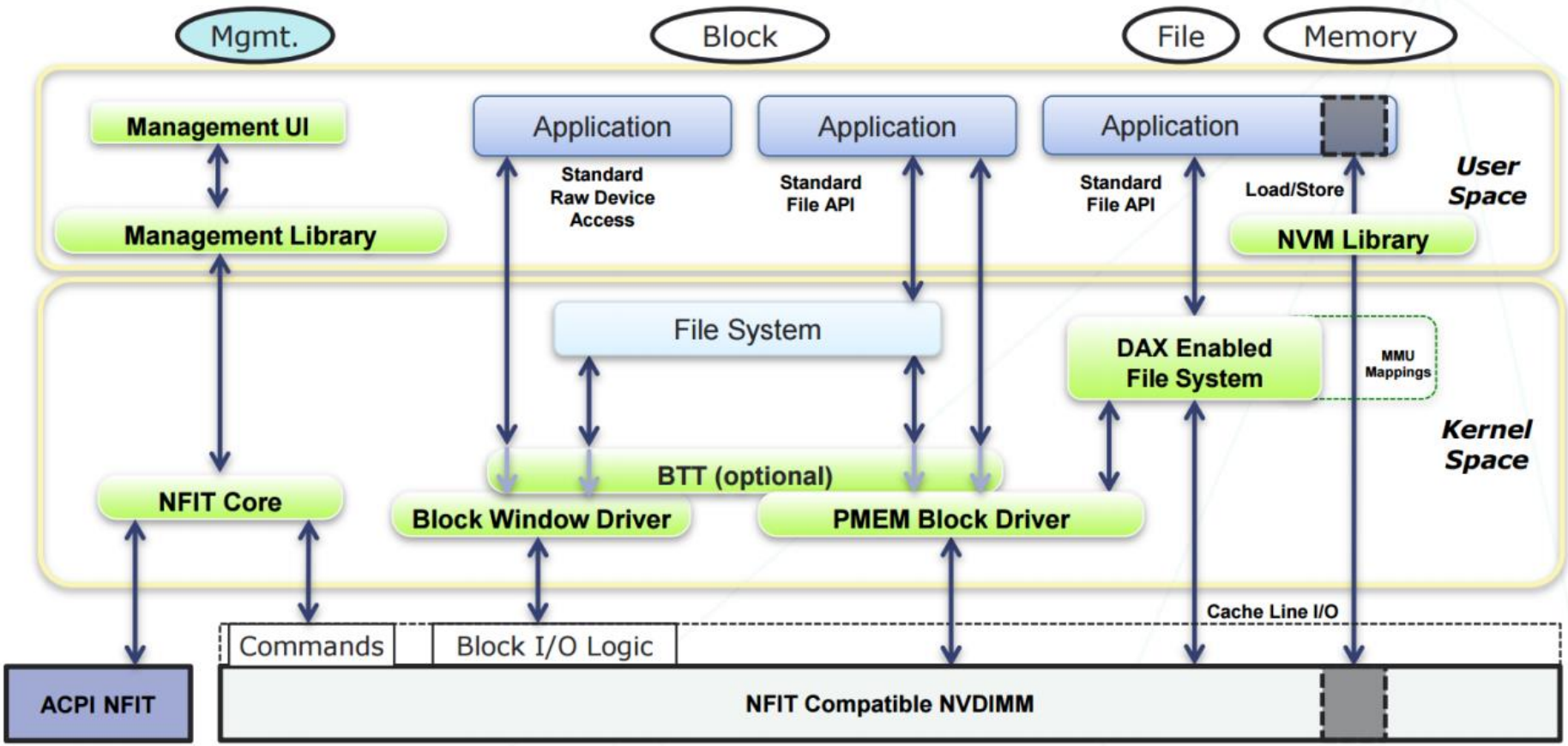
Early support of vNVDIMMs

ACPI 6.0 NFIT



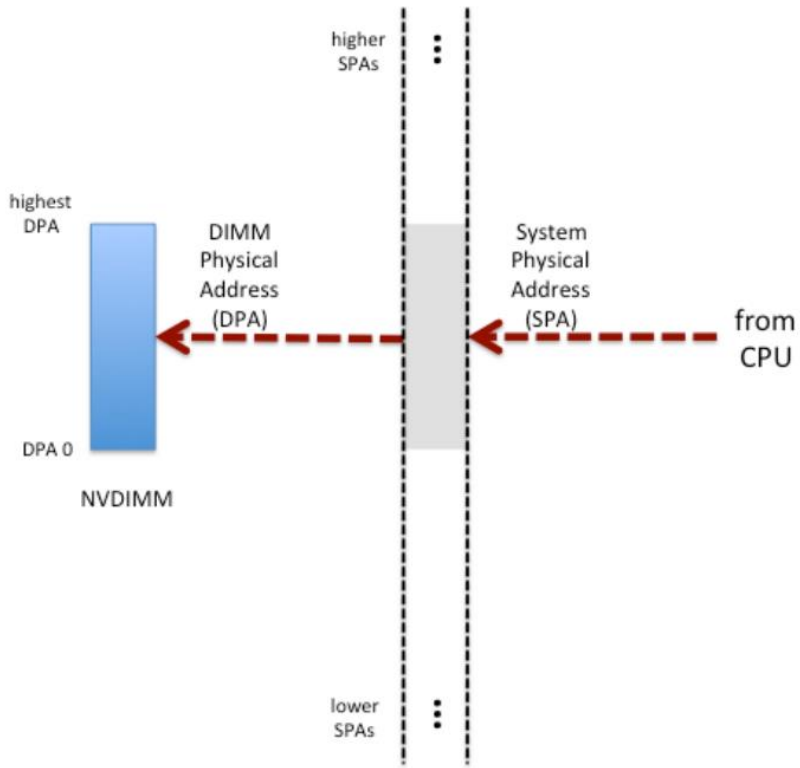
Picture taken from http://www.uefi.org/sites/default/files/resources/ACPI_6.0.pdf

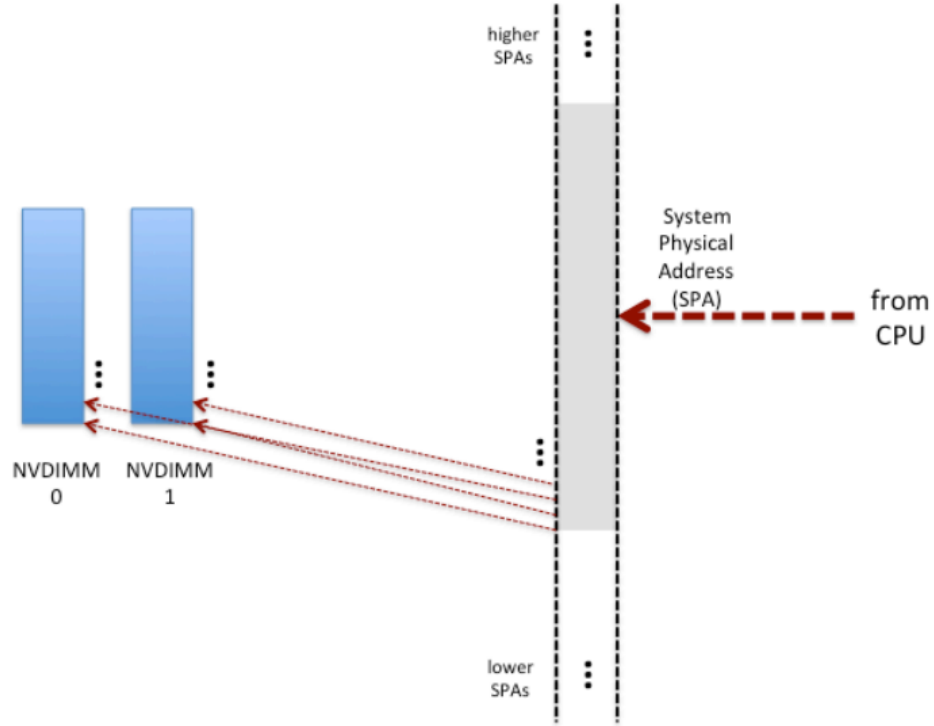
NVDIMM software architecture



Picture taken from http://pmem.io/documents/NVDIMM_Namespace_Spec.pdf

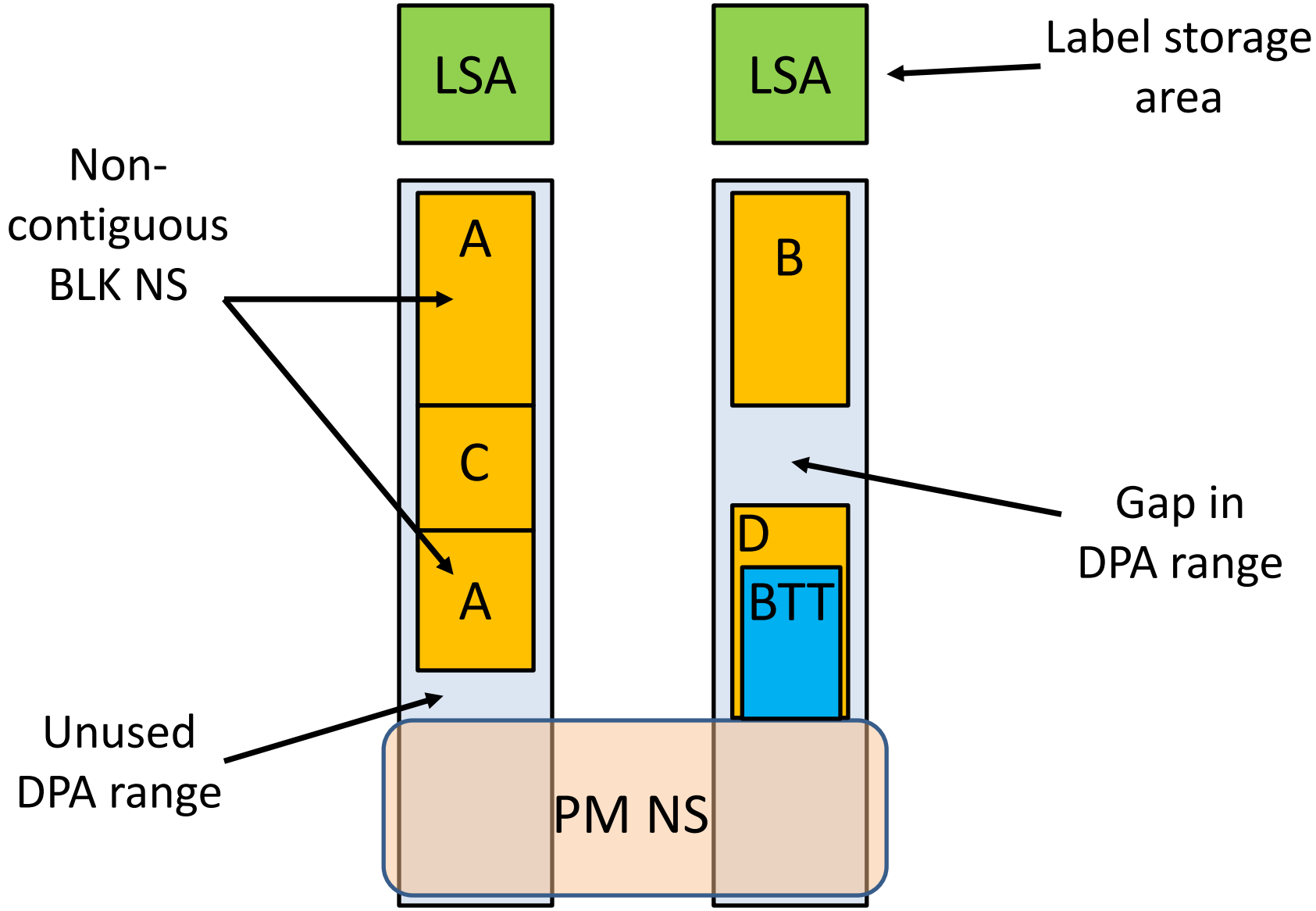
Memory interleaving



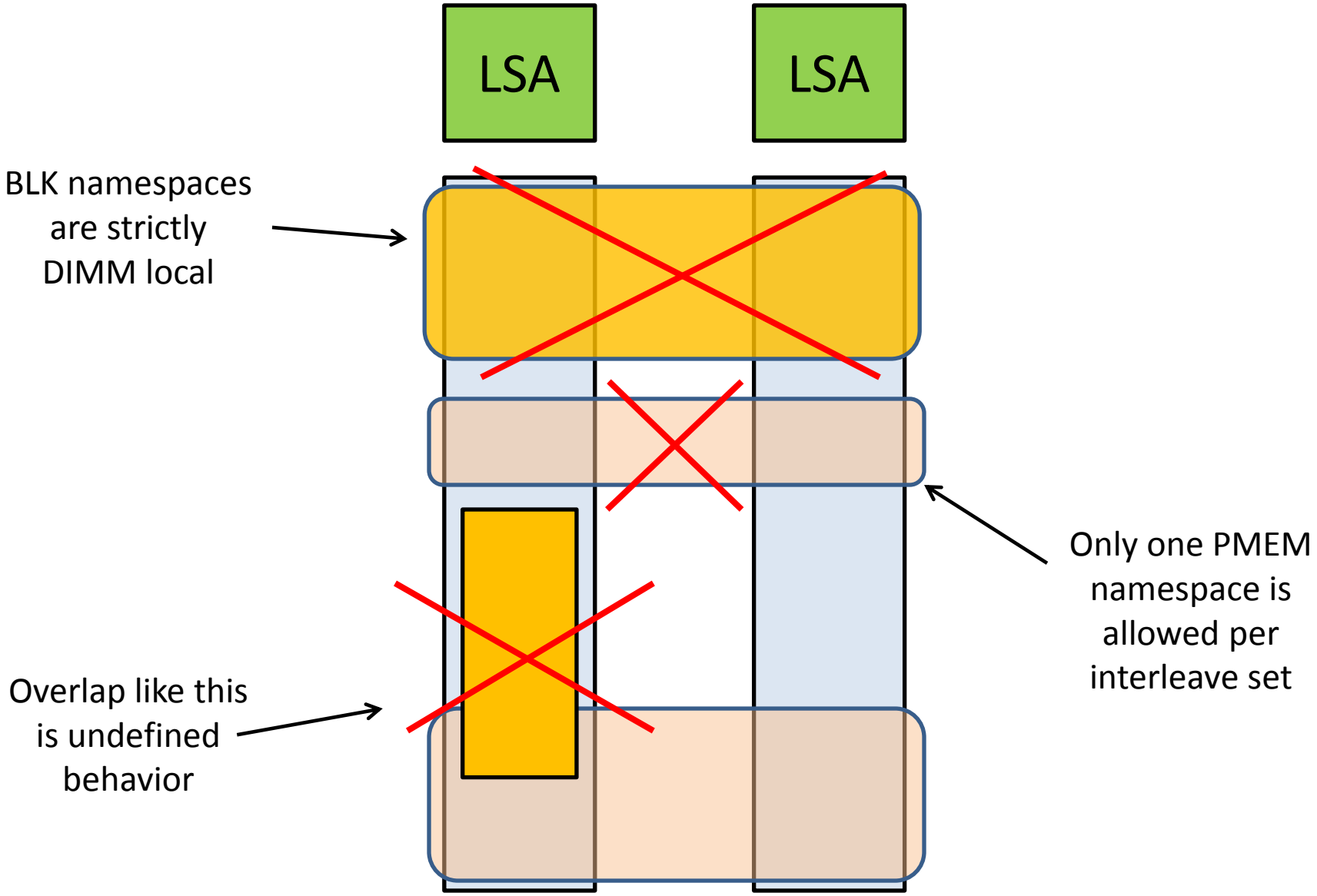


Pictures taken from http://pmem.io/documents/NVDIMM_Namespace_Spec.pdf

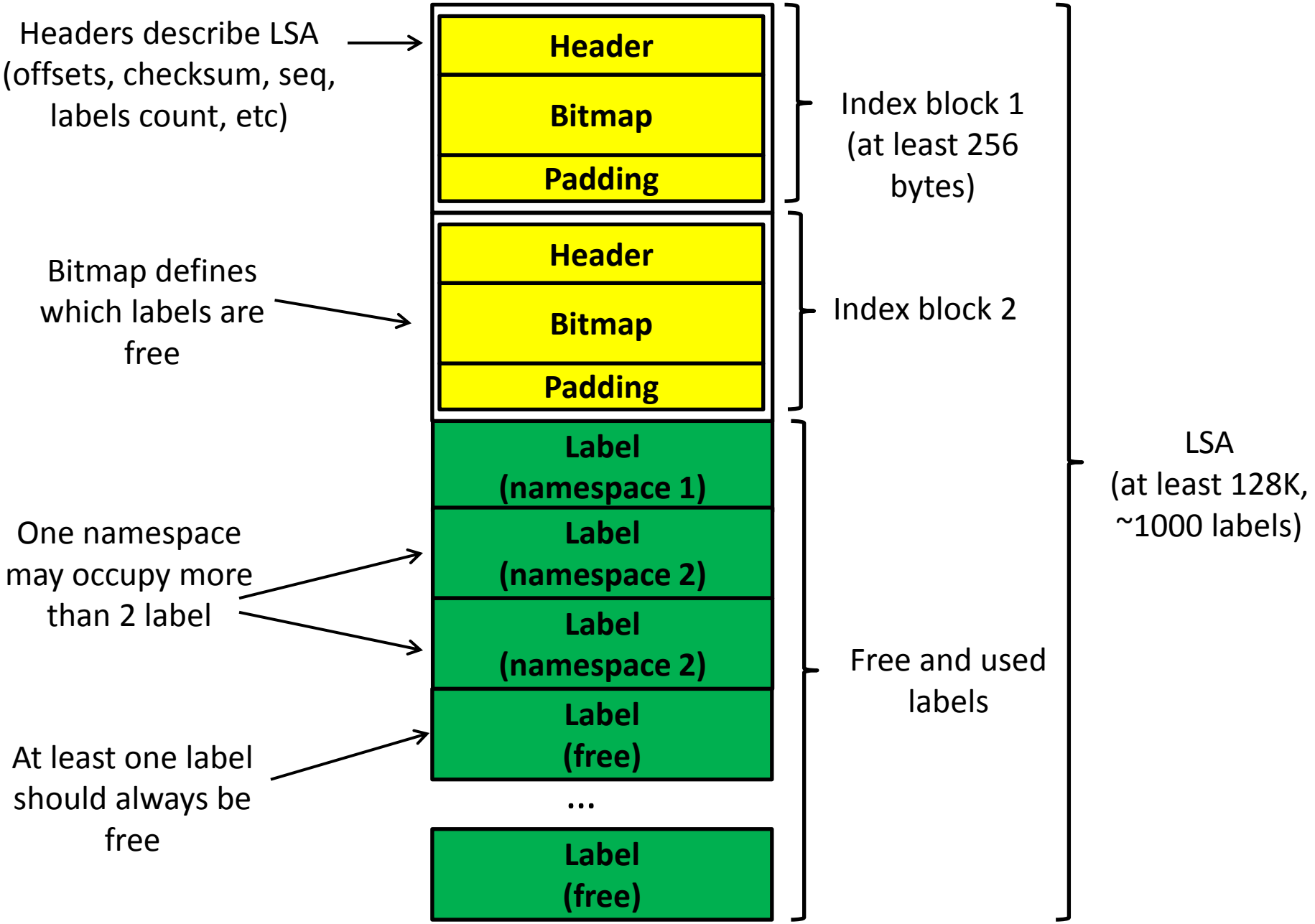
NVDIMM namespaces



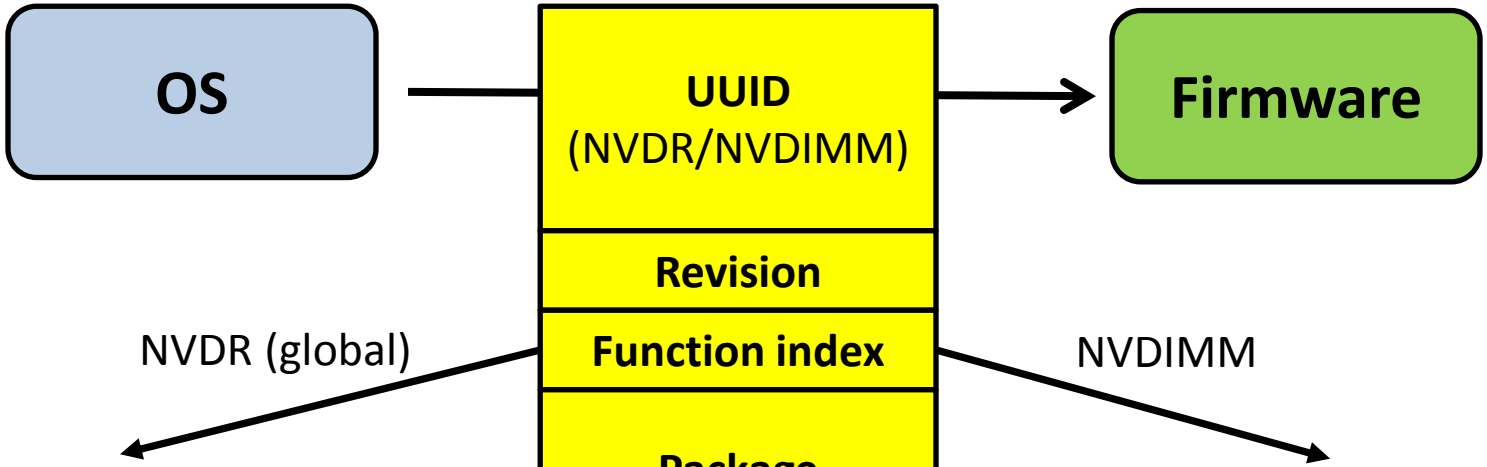
NVDIMM namespaces, basic rules



NVDIMM LSA


Headers describe LSA (offsets, checksum, seq, labels count, etc)
Header
Bitmap
Padding
Index block 1 (at least 256 bytes)
Bitmap defines which labels are free
Header
Bitmap
Padding
Index block 2
Label (namespace 1)
Label (namespace 2)
Label (namespace 2)
Label (free)
...
Label (free)
One namespace may occupy more than 2 label
At least one label should always be free
Free and used labels
LSA (at least 128K, ~1000 labels)

NVDIMM Intel DSM

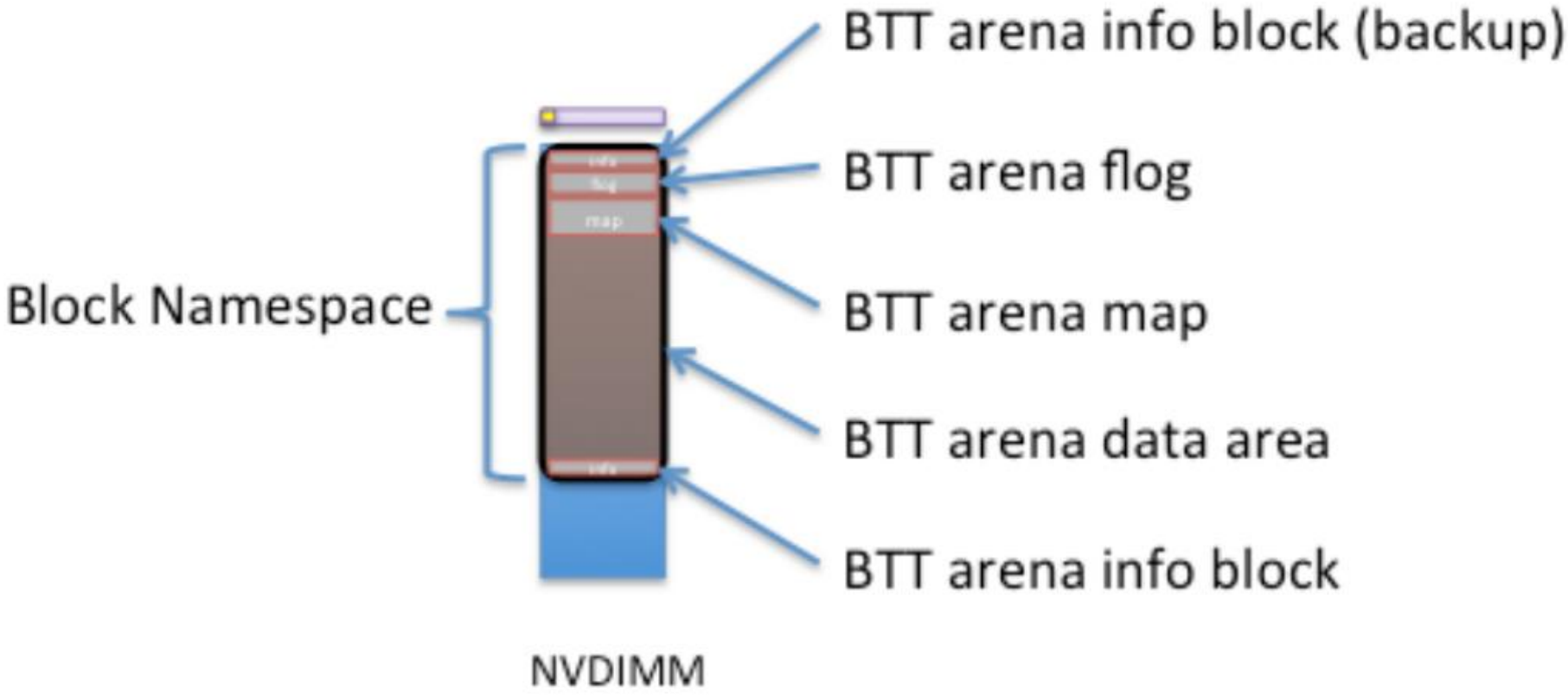


1. Query ARS caps
2. Start ARS
3. Query ARS

1. SMART & Health
2. Get SMART threshold
3. Get NVDIMM flags
4. Get LSA size
5. Get label data
6. Set label data
7. Get VSC effect log size
8. Get VSC effect log
9. VSC command

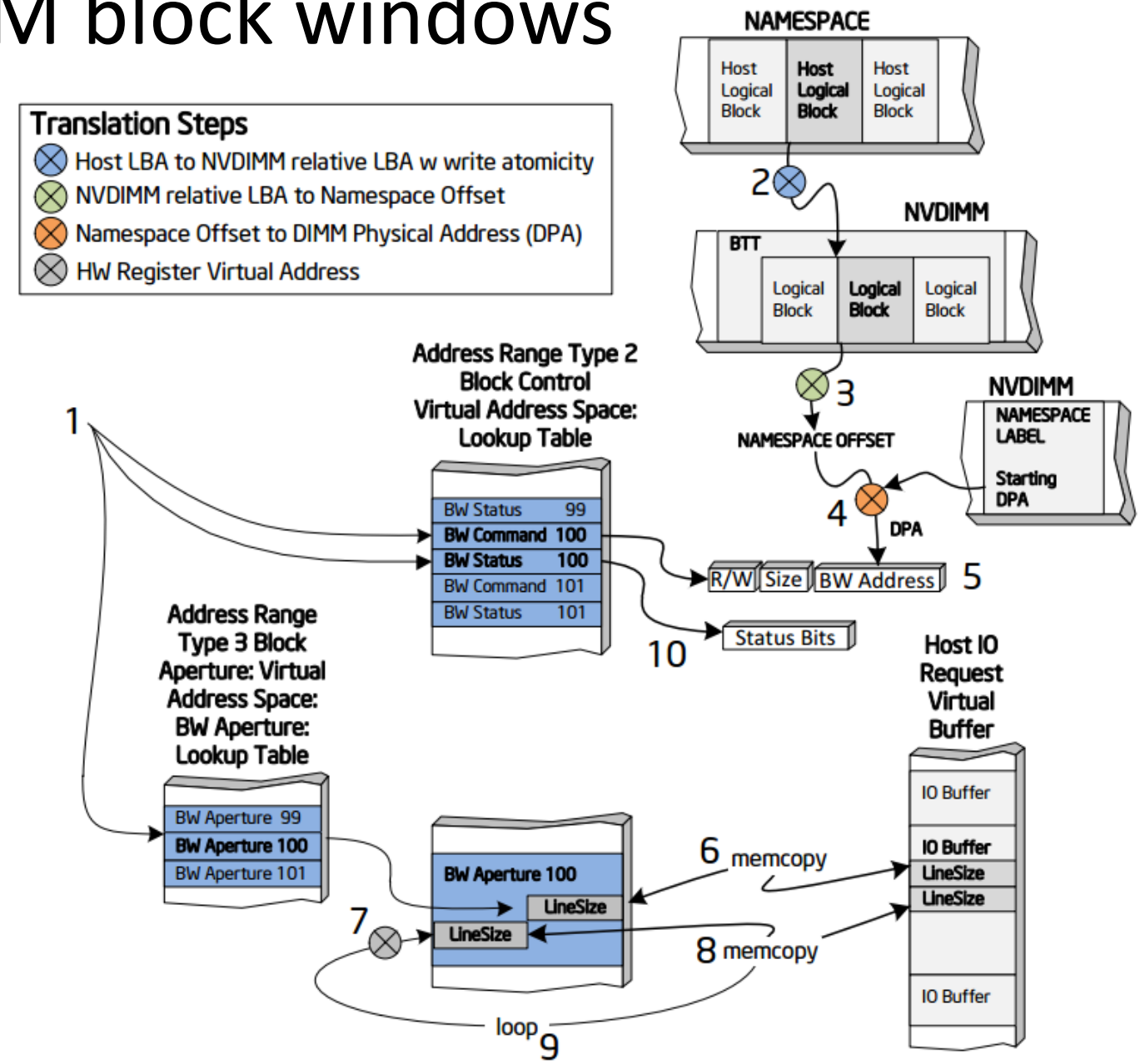
See http://pmem.io/documents/NVDIMM_DSM_Interface_Example.pdf for details

NVDIMM BTTs



Picture taken from http://pmem.io/documents/NVDIMM_Namespace_Spec.pdf

NVDIMM block windows

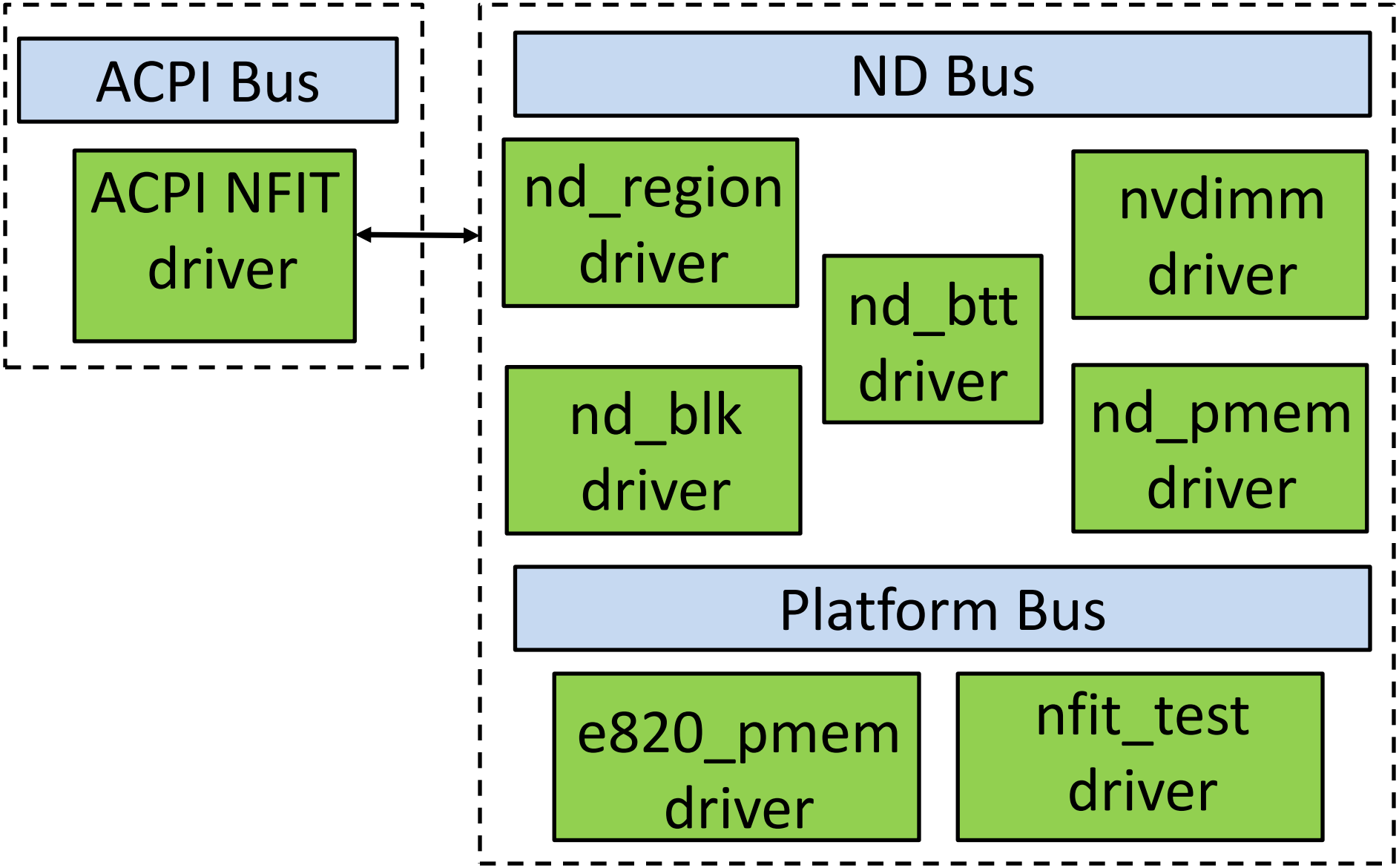


Picture taken from http://pmem.io/documents/NVDIMM_Driver_Writers_Guide.pdf

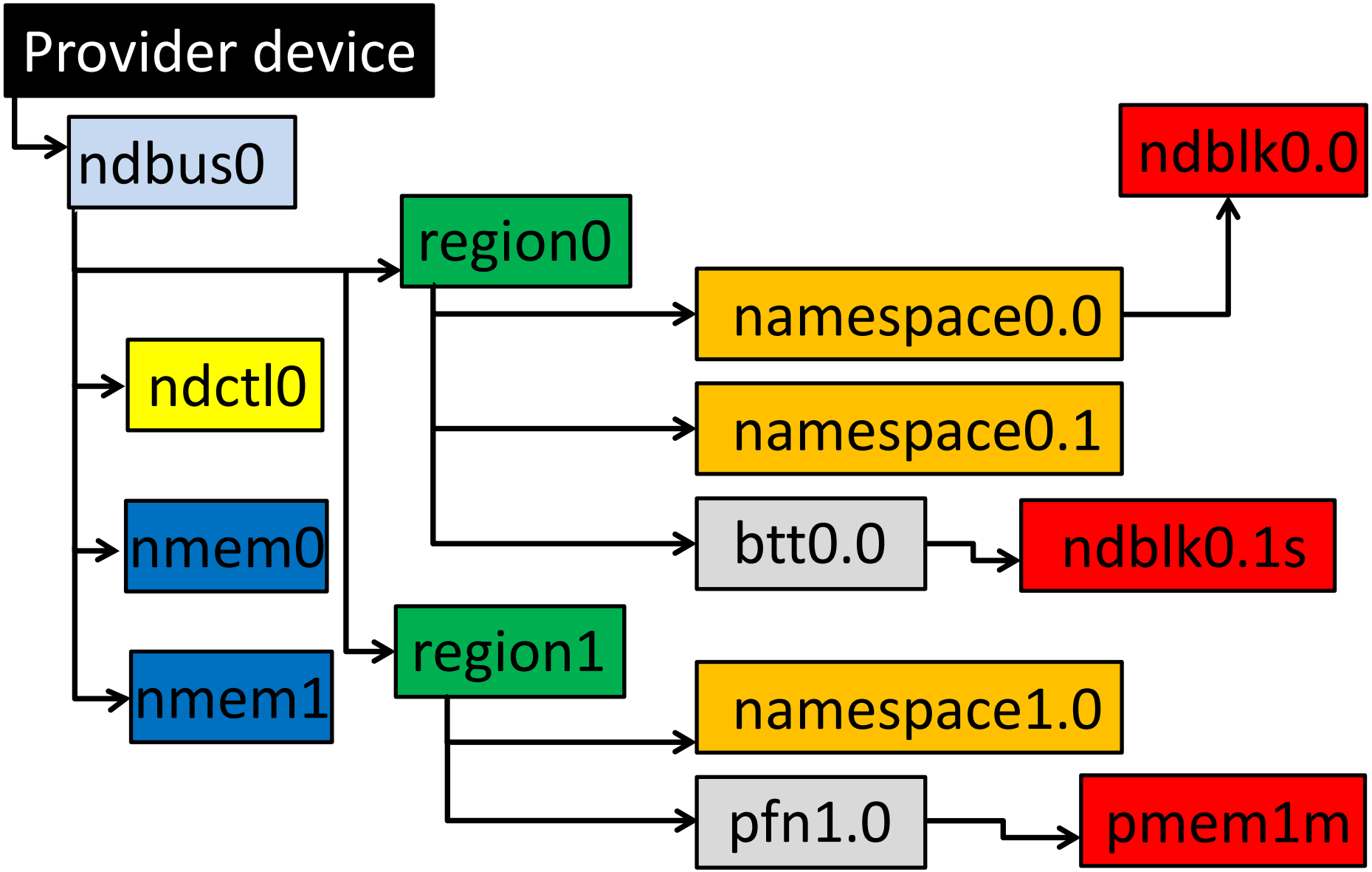
Linux kernel LIBND subsystem

- In mainline kernel since 4.2
- Actively developed
- Major pieces:
  - drivers/nvdimm
  - drivers/acpi/nfit.*
  - tools/testing/nvdimm
  - include/linux/pmem.h
  - Documentation/nvdimm
- Community
  - https://lists.01.org/mailman/listinfo/**linux-nvdimm**

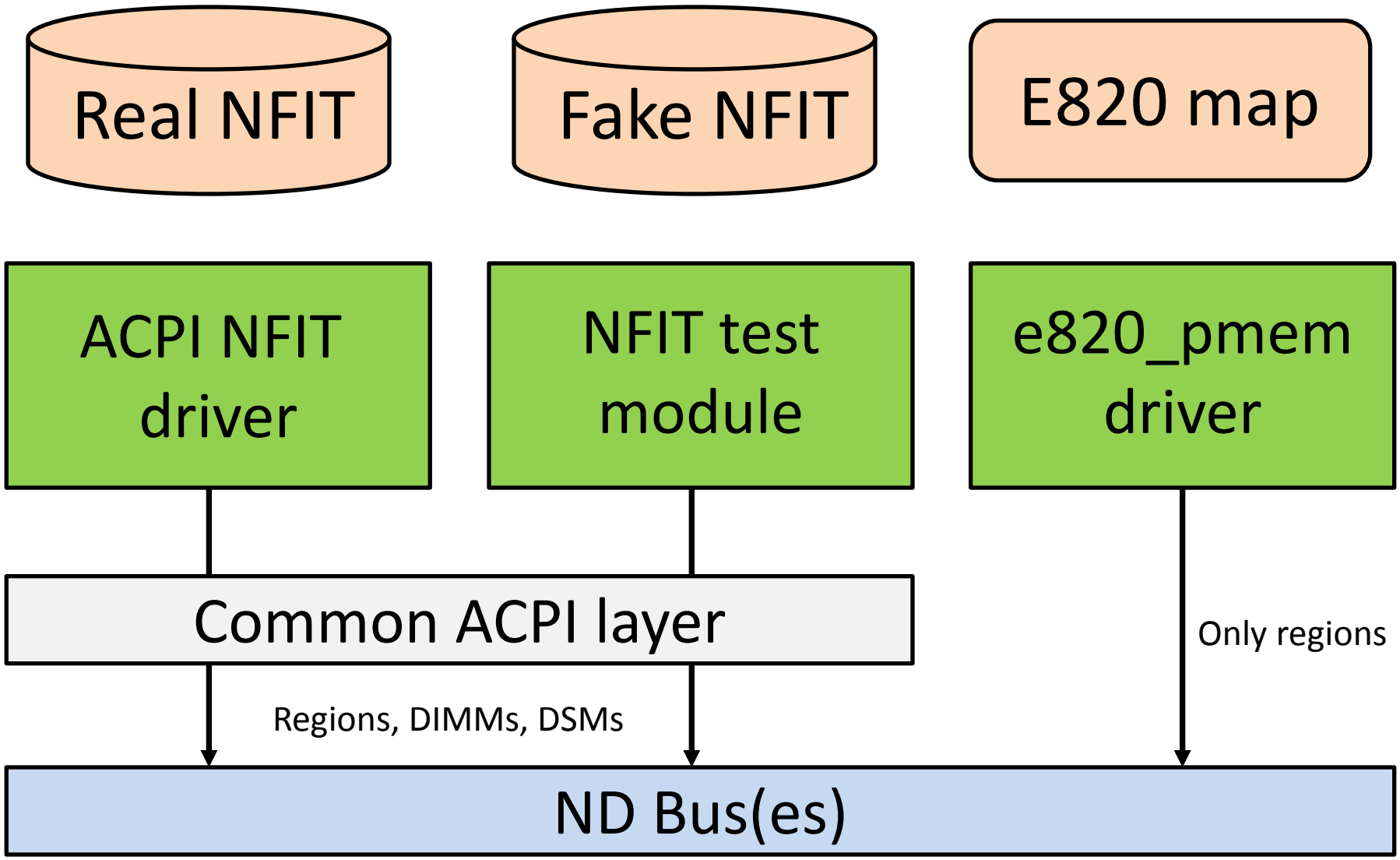
LIBND drivers



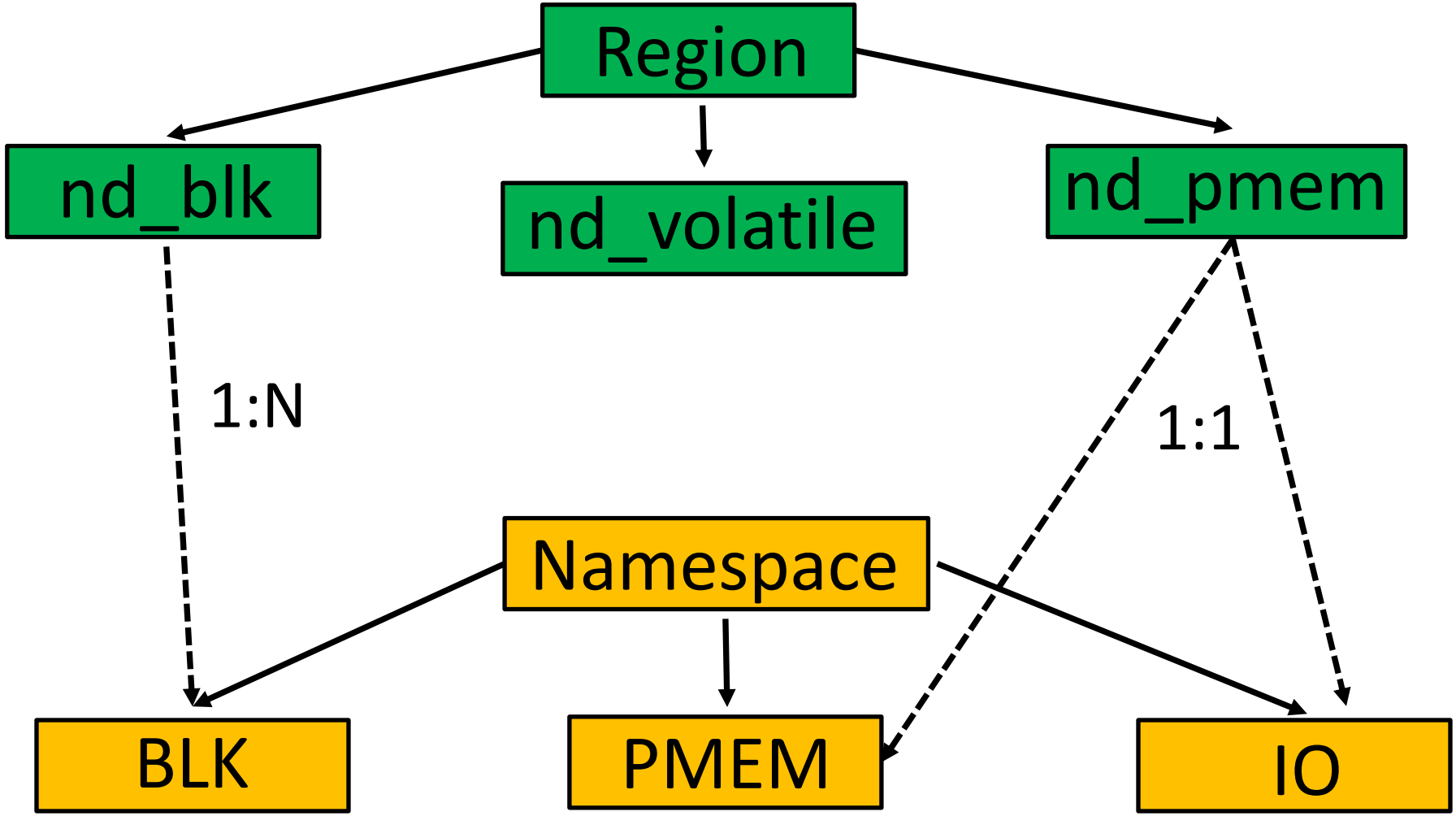
LIBND devices



LIBND bus providers (platform devices)



Region and namespace types



How to play with PMEM in Linux?

- Use ACPI 6.0 compatible NVDIMM hardware or legacy NVDIMMs
- Use virtual NVDIMMs provided by hypervisor
- RAM as persistent memory
- Use NFIT test module

Enable NVDIMM dynamic debug before you start playing with NVDIMMs

```
# echo "file *nvdimm* +p" > /sys/kernel/debug/dynamic_debug/control
```

vNVDIMM: QEMU emulation of NVDIMMs

- Not merged in QEMU mainline yet
  - https://github.com/xiaogr/qemu/tree/nvdimm-v7
  - Being reviewed on qemu-devel
- Supports PMEM only
- Exposes NFIT and only 3 subtables
- Good for development/testing with “real” NFIT

```
# qemu-system-x86_64 \
    -object memory-backend-file,share,id=mem1,mem-path=/dax/D1 \
    -device nvdimm,memdev=mem1,reserve-label-data,id=nv1  \
    -m 2048,maxmem=100G,slots=10 \
    <OTHER OPTIONS>
```

See docs/memory-hotplug.txt for details and syntax

NFIT test module

```
                    (a)                        (b)            DIMM   BLK-REGION
          +----------+--------------+----------+---------+
+------+  |  blk2.0  |     pm0.0    |  blk2.1  |  pm1.0  |    0      region2
| imc0 +--+- - - - - region0 - - - -+----------+         +
+--+---+  |  blk3.0  |     pm0.0    |  blk3.1  |  pm1.0  |    1      region3
   |      +----------+--------------v----------v         v
+--+---+                           |                    |
| cpu0 |                                  region1
+--+---+                           |                    |
   |      +-------------------------^----------^         ^
+--+---+  |                 blk4.0              |  pm1.0  |    2      region4
| imc1 +--+-------------------------+----------+         +
+------+  |                 blk5.0              |  pm1.0  |    3      region5
          +-------------------------+----------+-+-------+
```

```
+----------------------+
|----------------------|
||        pm2.0        ||
|----------------------|
+----------------------+
```

```
# make –C tools/testing/nvdimm install
# make modules_install
# modprobe nfit_test
```

You won’t be able to unload nfit_test immediately after loading it. You need to disable all regions/namespaces first via sysfs or ndctl.

RAM as PMEM via memmap

```
# cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-4.3.0-1-default root=UUID=39635fd6-64ee-
4538-9964-7de6bb181181 resume=/dev/sda1 splash=silent quiet
showopts memmap=1G!5G memmap=1G!7G
# dmesg | grep 'type 12'
[    0.000000] user: [mem 0x0000000140000000-0x000000017fffffff]
persistent (type 12)
[    0.000000] user: [mem 0x00000001c0000000-0x00000001ffffffff]
persistent (type 12)
#
```

- Each range is exposed as a new region under ndbus
- Always exactly one IO namespace per region
- BTT works (PFN should work as well)

ND buses in sysfs and devtmpfs

ND bus type

```
# ls /sys/bus/nd/
devices  drivers  drivers_autoprobe  drivers_probe  uevent
# find /sys/ -name ndbus*
/sys/devices/platform/e820_pmem/ndbus0
/sys/devices/platform/nfit_test.0/ndbus1
/sys/devices/platform/nfit_test.1/ndbus2
#
```

Provider device

Bus name

Char device per ND bus for IOCTLs

```
# ls -l /dev/ndctl*
crw------- 1 root root 250, 0 Nov 15 06:15 /dev/ndctl0
crw------- 1 root root 250, 1 Nov 15 06:16 /dev/ndctl1
crw------- 1 root root 250, 2 Nov 15 06:16 /dev/ndctl2
#
```

NVDIMMs in sysfs and devtmpfs

ND bus name

```
# ls -l /sys/bus/nd/devices/ | grep nmem | lsshrink
lrwxrwxrwx nmem0 -> ../../../devices/platform/nfit_test.0/ndbus1/nmem0
lrwxrwxrwx nmem1 -> ../../../devices/platform/nfit_test.0/ndbus1/nmem1
lrwxrwxrwx nmem2 -> ../../../devices/platform/nfit_test.0/ndbus1/nmem2
lrwxrwxrwx nmem3 -> ../../../devices/platform/nfit_test.0/ndbus1/nmem3
lrwxrwxrwx nmem4 -> ../../../devices/platform/nfit_test.1/ndbus2/nmem4
#
```

One nmemX per NVDIMM

Provider device

Char device per NVDIMM for IOCTLs

```
# ls -l /dev/nmem*
crw------- 1 root root 249, 0 Nov 15 06:16 /dev/nmem0
crw------- 1 root root 249, 1 Nov 15 06:16 /dev/nmem1
crw------- 1 root root 249, 2 Nov 15 06:16 /dev/nmem2
crw------- 1 root root 249, 3 Nov 15 06:16 /dev/nmem3
crw------- 1 root root 249, 4 Nov 15 06:16 /dev/nmem4
#
```

NVDIMMs sysfs attributes

```
# ls -l /sys/devices/platform/nfit_test.0/ndbus1/nmem0/ | lsshrink
total
-r--r--r-- available_slots
-r--r--r-- commands
-r--r--r-- dev
-r--r--r-- devtype
lrwxrwxrwx driver -> ../../../../../bus/nd/drivers/nvdimm
-r--r--r-- modalias
drwxr-xr-x nfit
drwxr-xr-x power
-r--r--r-- state
lrwxrwxrwx subsystem -> ../../../../../bus/nd
-rw-r--r-- uevent
# cat /sys/devices/platform/nfit_test.0/ndbus1/nmem0/available_slots
1015
# cat /sys/devices/platform/nfit_test.0/ndbus1/nmem0/commands
get_size get_data set_data
# cat /sys/devices/platform/nfit_test.1/ndbus2/nmem4/nfit/flags
save_fail restore_fail flush_fail not_armed smart_event
#
```

If NVDIMM is reported as *not armed*, then its regions will be marked read-only. State will be *idle* for NVDIMMs w/o aliased regions and *active* otherwise.

Regions in sysfs

Region numbers space is global, not bus local

```
# ls -l /sys/bus/nd/devices/ | grep 'region[0-9] ' | lsshrink
lrwxrwxrwx region0 -> ../../../devices/platform/e820_pmem/ndbus0/region0
lrwxrwxrwx region1 -> ../../../devices/platform/e820_pmem/ndbus0/region1
lrwxrwxrwx region2 -> ../../../devices/platform/nfit_test.0/ndbus1/region2
lrwxrwxrwx region3 -> ../../../devices/platform/nfit_test.0/ndbus1/region3
lrwxrwxrwx region4 -> ../../../devices/platform/nfit_test.0/ndbus1/region4
lrwxrwxrwx region5 -> ../../../devices/platform/nfit_test.0/ndbus1/region5
lrwxrwxrwx region6 -> ../../../devices/platform/nfit_test.0/ndbus1/region6
lrwxrwxrwx region7 -> ../../../devices/platform/nfit_test.0/ndbus1/region7
lrwxrwxrwx region8 -> ../../../devices/platform/nfit_test.1/ndbus2/region8
#
```

Regions

Regions live under ND buses

Note that region names (numbers) depend on bus initialization order and other factors. So regions and all dependent objects (NS, BTT, PNF) should be addressed via stable names (e.g. UUID derived).

Regions in sysfs (BLK)

```
# cd /sys/devices/platform/nfit_test.0/ndbus1/region4
# ls -l | lsshrink
total
-r--r--r-- available_size
drwxr-xr-x btt4.0
-r--r--r-- btt_seed
-r--r--r-- devtype
lrwxrwxrwx driver -> ../../../../../bus/nd/drivers/nd_region
-r--r--r-- init_namespaces
-r--r--r-- mapping0
-r--r--r-- mappings
-r--r--r-- modalias
drwxr-xr-x namespace4.0
-r--r--r-- namespace_seed
drwxr-xr-x nfit
-r--r--r-- nstype
-r--r--r-- numa_node
-r--r--r-- pfn_seed
drwxr-xr-x power
-rw-r--r-- read_only
-r--r--r-- size
lrwxrwxrwx subsystem -> ../../../../../bus/nd
-rw-r--r-- uevent
#
```

Regions in sysfs (PMEM)

```
# cd /sys/devices/platform/nfit_test.0/ndbus1/region2
# ls -l | lsshrink
total
-r--r--r-- available_size
drwxr-xr-x btt2.0
-r--r--r-- btt_seed
-r--r--r-- devtype
lrwxrwxrwx driver -> ../../../../../bus/nd/drivers/nd_region
-r--r--r-- init_namespaces
-r--r--r-- mapping0
-r--r--r-- mapping1
-r--r--r-- mappings
-r--r--r-- modalias
drwxr-xr-x namespace2.0
-r--r--r-- namespace_seed
drwxr-xr-x nfit
-r--r--r-- nstype
-r--r--r-- numa_node
-r--r--r-- pfn_seed
drwxr-xr-x power
-rw-r--r-- read_only
-r--r--r-- set_cookie
-r--r--r-- size
lrwxrwxrwx subsystem -> ../../../../../bus/nd
-rw-r--r-- uevent
#
```

Namespaces in sysfs (BLK NS)

```
# cd /sys/devices/platform/nfit_test.0/ndbus1/region4/namespace4.0/
# ls -l | lsshrink
total
-rw-r--r-- alt_name
-r--r--r-- devtype
-r--r--r-- dpa_extents
-rw-r--r-- force_raw
-r--r--r-- holder
-r--r--r-- modalias
-r--r--r-- nstype
-r--r--r-- numa_node
drwxr-xr-x power
-rw-r--r-- sector_size
-rw-r--r-- size
lrwxrwxrwx subsystem -> ../../../../../../bus/nd
-rw-r--r-- uevent
-rw-r--r-- uuid
#
```

Namespace size should be at least 4MB, but if you want to create BTT, then hosting namespace size should be at least 16MB.

Namespaces in sysfs (PMEM NS)

```
# cd /sys/devices/platform/nfit_test.0/ndbus1/region3/namespace3.0/
# ls -l | lsshrink
total
-rw-r--r-- alt_name
-r--r--r-- devtype
-r--r--r-- dpa_extents
-rw-r--r-- force_raw
-r--r--r-- holder
-r--r--r-- modalias
-r--r--r-- nstype
-r--r--r-- numa_node
drwxr-xr-x power
-r--r--r-- resource
-rw-r--r-- size
lrwxrwxrwx subsystem -> ../../../../../../bus/nd
-rw-r--r-- uevent
-rw-r--r-- uuid
#
```

PMEM namespace size must be 4K*mappings aligned (e.g. 8K for 2 way IS).
DAX is not possible on BTT on top of PMEM, only on PMEM itself.

Create namespace via sysfs

```
# cd /sys/devices/platform/nfit_test.0/ndbus1/region4
# cat namespace_seed
namespace4.0
# cd namespace4.0
# lsblk | grep ndblk
# uuidgen > uuid
# echo "My namespace" > alt_name
# echo 512 > sector_size
# echo 10485760 > size
# echo namespace4.0 > /sys/bus/nd/drivers/nd_blk/bind
# lsblk | grep ndblk
ndblk4.0 259:3    0     10M  0 disk
# cat ../size
33554432
# cat ../available_size
23068672
# echo namespace4.0 > /sys/bus/nd/drivers/nd_blk/unbind
# lsblk | grep ndblk
#
```

Be careful with changing size of unbound namespace. You can easily shrink it and corrupt the file system like this:

EXT4-fs (pmem0): bad geometry: block count 15360 exceeds size of device (10240 blocks)

For PMEM, check if seed namespace was instantiated already first.

Create BTT via sysfs

```
# cd /sys/devices/platform/nfit_test.0/ndbus1/region5
# cat namespace_seed
namespace5.0
# cd namespace5.0
# uuidgen > uuid
# echo "BTT holder" > alt_name
# echo 20971520 > size
# echo 4096 > sector_size
# lsblk | grep ndblk
# cd ../
# cat btt_seed
btt5.0
# cd btt5.0
# uuidgen > uuid
# echo 512 > sector_size
# echo namespace5.0 > namespace
# echo btt5.0 > /sys/bus/nd/drivers/nd_blk/bind
# lsblk | grep ndblk
ndblk5.0s 259:3    0   19.7M  0 disk
# cat ../namespace5.0/holder
btt5.0
#
```

You can't unbind BTT, *echo "" > namespace* and then bind the namespace. BTT will be bound instead. Use *echo 1 > force_raw* in this case.

Manage NVDIMMs via libndctl

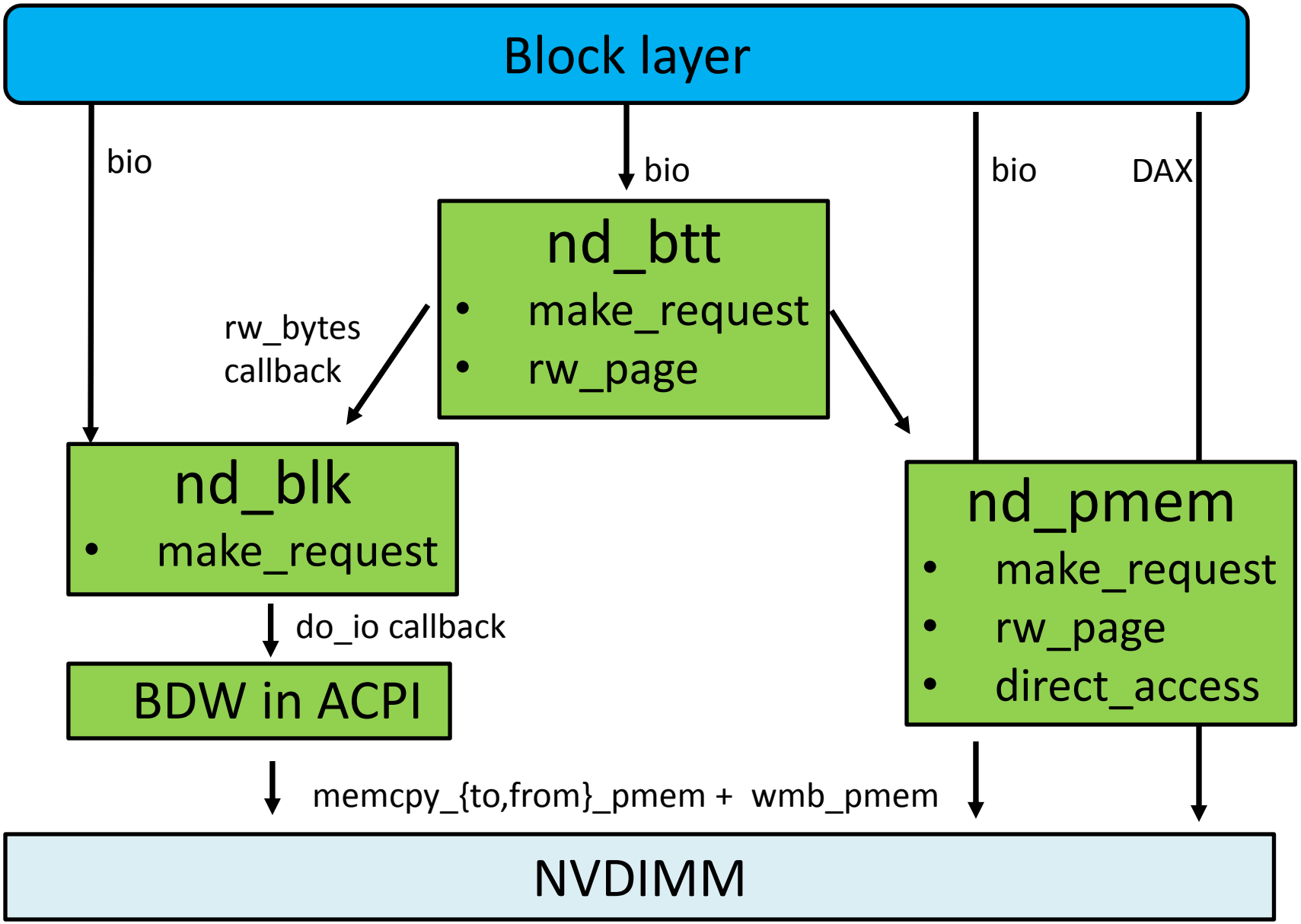
https://github.com/pmem/ndctl

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <ndctl/libndctl.h>
4
5 int
6 main ()
7 {
8   struct ndctl_ctx *ctx;
9   struct ndctl_bus *bus;
10  struct ndctl_region *region;
11
12  if (ndctl_new (&ctx) < 0)
13    exit (EXIT_FAILURE);
14
15  ndctl_bus_foreach (ctx, bus)
16  {
17    ndctl_region_foreach (bus, region)
18    {
19      printf ("BUS=%-10s Name=%-12s Type=%-10s Size=%-12llu Available=%-12llu\n",
20              ndctl_bus_get_devname (bus), ndctl_region_get_devname (region),
21              ndctl_region_get_type_name (region),
22              ndctl_region_get_size (region),
23              ndctl_region_get_available_size (region));
24    }
25  }
26  exit (EXIT_SUCCESS);
27 }
```

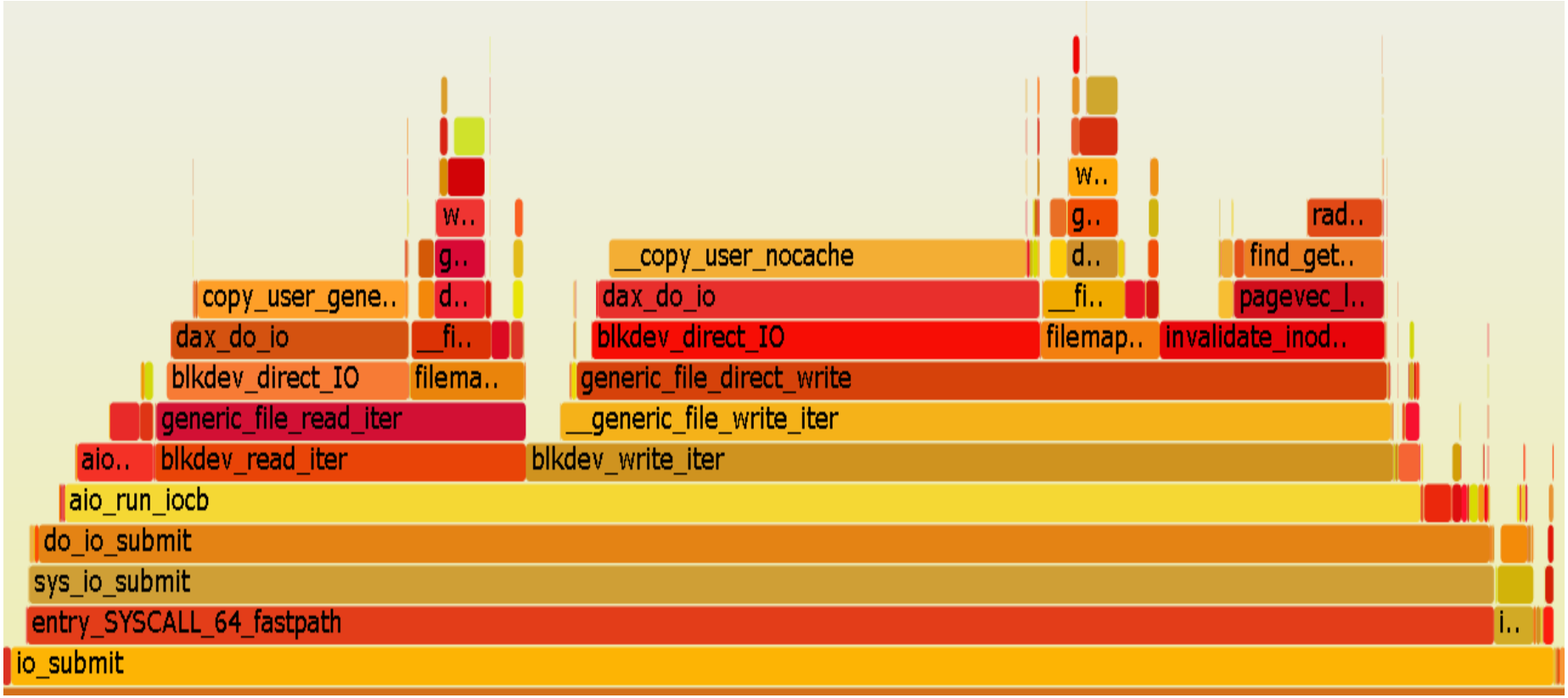
Manage NVDIMMs via libndctl, part 2

```
$ gcc -Wall -Werror -lndctl  -o ./dump_regions ./dump_regions.c
$ sudo ./dump_regions
BUS=ndbus2    Name=region8    Type=pmem    Size=33554432    Available=0
BUS=ndbus1    Name=region7    Type=blk     Size=33554432    Available=33554432
BUS=ndbus1    Name=region6    Type=blk     Size=33554432    Available=33554432
BUS=ndbus1    Name=region5    Type=blk     Size=33554432    Available=33554432
BUS=ndbus1    Name=region4    Type=blk     Size=33554432    Available=33554432
BUS=ndbus1    Name=region3    Type=pmem    Size=67108864    Available=67108864
BUS=ndbus1    Name=region2    Type=pmem    Size=33554432    Available=33554432
BUS=ndbus0    Name=region1    Type=pmem    Size=1073741824  Available=0
BUS=ndbus0    Name=region0    Type=pmem    Size=1073741824  Available=0
$
```

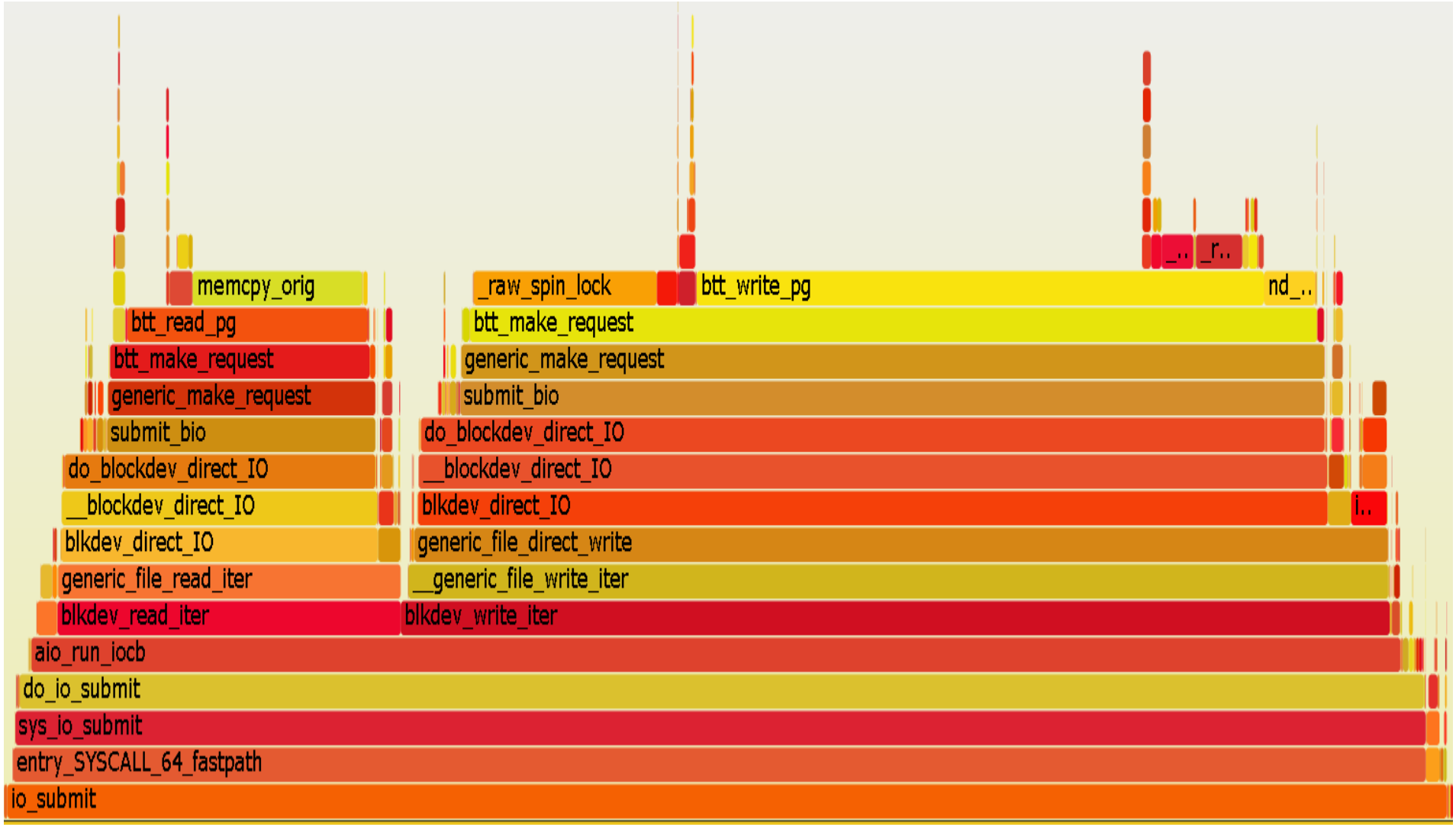
NVDIMM block I/O path



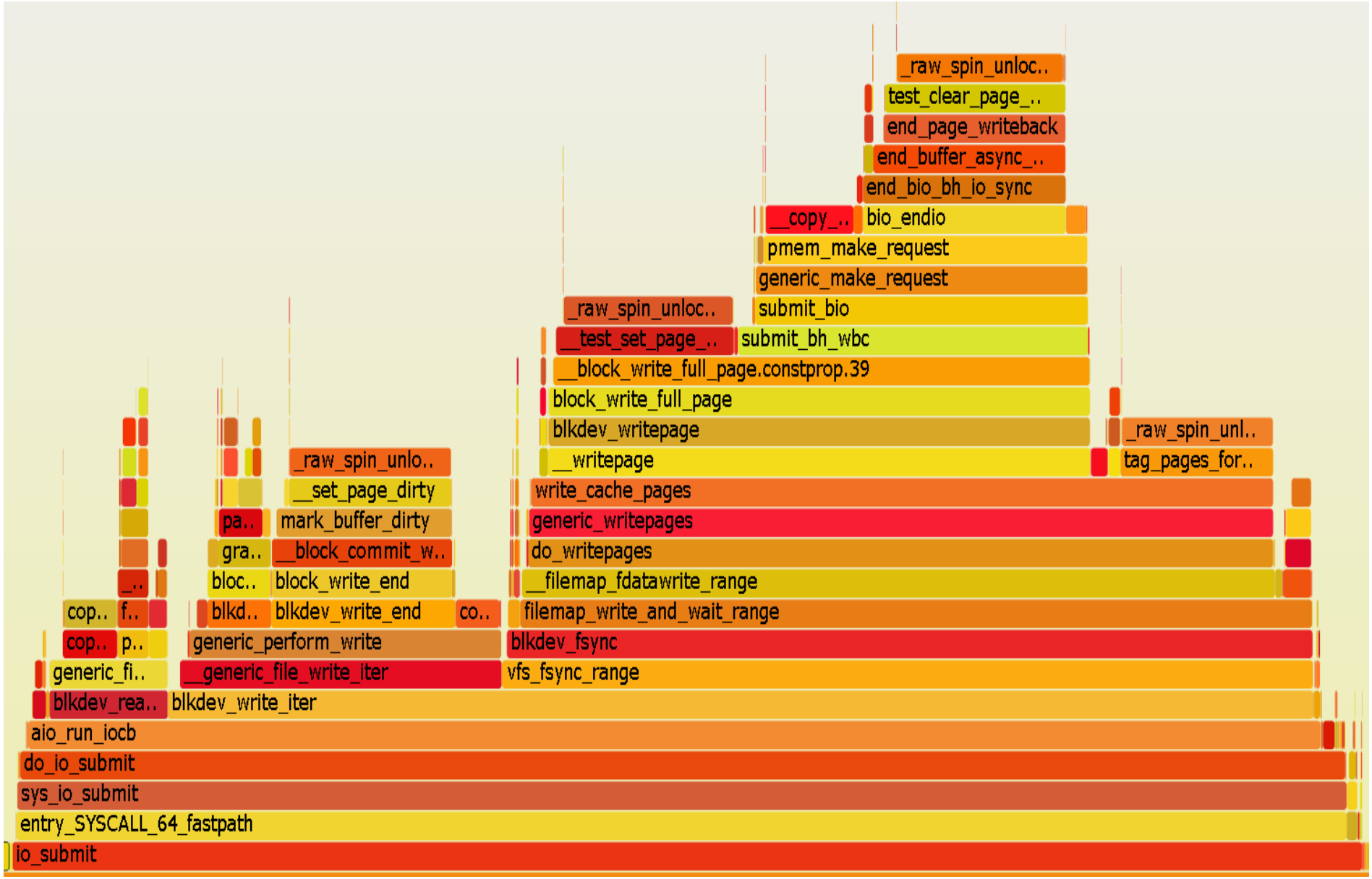
PMEM direct I/O



BTT over PMEM direct I/O



PMEM sync I/O


_raw_spin_unloc..
test_clear_page_..
end_page_writeback
end_buffer_async_..
end_bio_bh_io_sync
__copy_..
bio_endio
pmem_make_request
generic_make_request
_raw_spin_unloc..
submit_bio
__test_set_page_..
submit_bh_wbc
__block_write_full_page.constprop.39
block_write_full_page
blkdev_writepage
_raw_spin_unl..
_raw_spin_unlo..
__writepage
tag_pages_for..
__set_page_dirty
write_cache_pages
pa..
mark_buffer_dirty
generic_writepages
gra..
__block_commit_w..
do_writepages
_..
bloc..
block_write_end
__filemap_fdatawrite_range
cop..
f..
blkd..
blkdev_write_end
co..
filemap_write_and_wait_range
cop..
p..
generic_perform_write
blkdev_fsync
generic_fi..
__generic_file_write_iter
vfs_fsync_range
blkdev_rea..
blkdev_write_iter
aio_run_iocb
do_io_submit
sys_io_submit
entry_SYSCALL_64_fastpath
io_submit

Block I/O performance

```
linux-pb7x:~ # ./fio ./pmem_btt.fio
rand-rw: (g=0): rw=randrw, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=1
...
fio-2.2.8
Starting 16 processes
Jobs: 16 (f=16): [m(16)] [5.0% done] [3457MB/0KB/0KB /s] [885K/0/0 iops] [e
```

BTT cost

```
linux-pb7x:~ # ./fio ./pmem.fio
rand-rw: (g=0): rw=randrw, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=1
...
fio-2.2.8
Starting 16 processes
Jobs: 16 (f=16): [m(16)] [18.8% done] [9034MB/0KB/0KB /s] [2313K/0/0 iops]
```

Direct vs sync

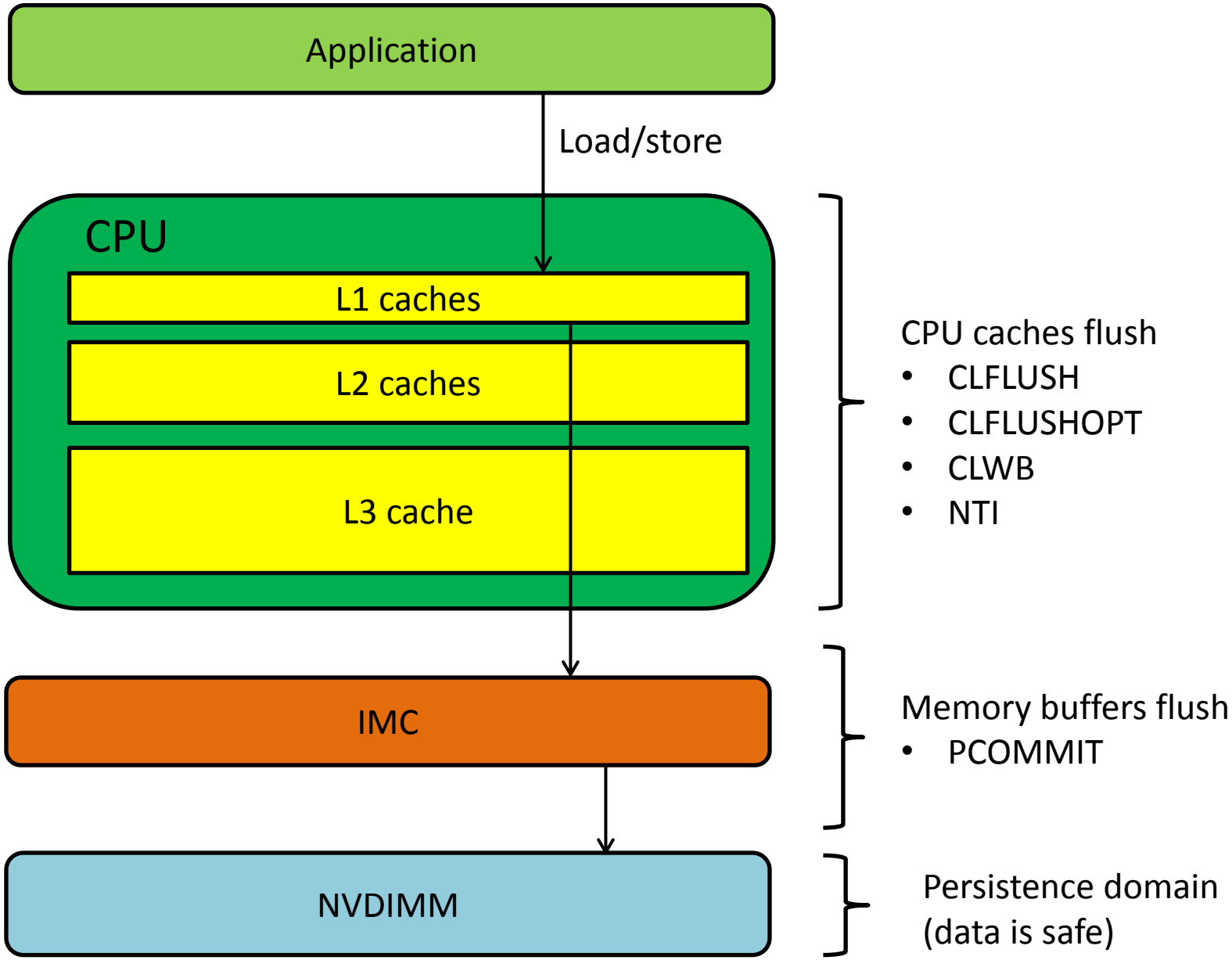
```
linux-pb7x:~ # ./fio ./pmem_sync.fio
rand-rw: (g=0): rw=randrw, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=1
...
fio-2.2.8
Starting 16 processes
Jobs: 16 (f=16): [m(16)] [5.9% done] [2808MB/0KB/0KB /s] [719K/0/0 iops] [e
```

DAX

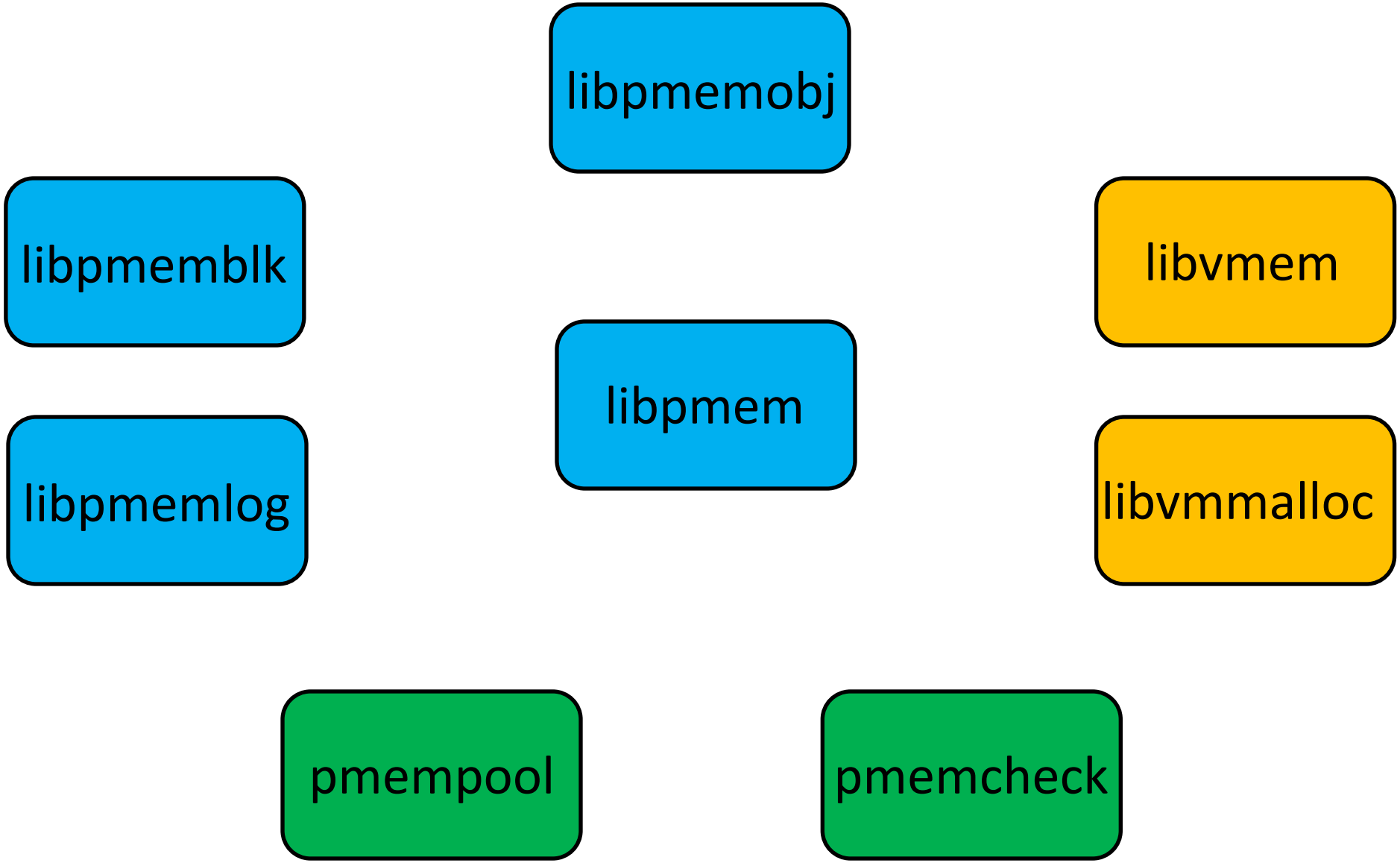
```
# mkfs.ext4 -b 4096 /dev/pmem1
mke2fs 1.42.11 (09-Jul-2014)
Creating filesystem with 262144 4k blocks and 65536 inodes
Filesystem UUID: 40716a1a-9360-4d8e-8300-a70bb844d048
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

# mount -t ext4 -o dax /dev/pmem1 /tmp/dax/
#
```

- Only ext2, ext4 and xfs currently support DAX
- Note that block size should match page size
- Still kind of experimental (at least in xfs)

PMEM I/O path (kernel bypass)



NVM Library, because pmem is not easy…



See http://pmem.io/nvml/ and https://github.com/pmem/nvml

Q & A

Thanks!

Backup slides

Sample NFIT dump

```
Signature : "NFIT"    [NVDIMM Firmware Interface Table]
[004h 0004   4]                 Table Length : 000000E0
[008h 0008   1]                     Revision : 01
[009h 0009   1]                     Checksum : 79
[00Ah 0010   6]                       Oem ID : "BOCHS "
[010h 0016   8]                 Oem Table ID : "BXPCNFIT"
[018h 0024   4]                 Oem Revision : 00000001
[01Ch 0028   4]              Asl Compiler ID : "BXPC"
[020h 0032   4]        Asl Compiler Revision : 00000001

[024h 0036   4]                     Reserved : 00000000

[028h 0040   2]                 Subtable Type : 0000 [System Physical Address Range]
[02Ah 0042   2]                       Length : 0038

[02Ch 0044   2]                  Range Index : 0002
[02Eh 0046   2]        Flags (decoded below) : 0003
                   Add/Online Operation Only : 1
                      Proximity Domain Valid : 1
[030h 0048   4]                     Reserved : 00000000
[034h 0052   4]             Proximity Domain : 00000000
[038h 0056  16]           Address Range GUID : 66F0D379-B4F3-4074-AC43-0D3318B78CDB
[048h 0072   8]           Address Range Base : 0000000100000000
[050h 0080   8]         Address Range Length : 00000000FFFE0000
[058h 0088   8]         Memory Map Attribute : 0000000000008008

[060h 0096   2]                 Subtable Type : 0001 [Memory Range Map]
[062h 0098   2]                       Length : 0030

[064h 0100   4]                Device Handle : 00000001
[068h 0104   2]                  Physical Id : 0000
[06Ah 0106   2]                    Region Id : 0000
[06Ch 0108   2]                  Range Index : 0002
[06Eh 0110   2]         Control Region Index : 0003
[070h 0112   8]                  Region Size : 00000000FFFE0000
[078h 0120   8]                Region Offset : 0000000000000000
[080h 0128   8]          Address Region Base : 0000000100000000
```