Supporting zoned block devices with non-power-of-2 zone sizes

LPC 2022

Pankaj Raghav,
Software Developer at
Samsung Semiconductor Denmark Research

Agenda

- Zoned block device support in Linux: Past & Present
- Non-power-of-2 zone size support in Linux
- Conclusion and Future work

Part 1

Zoned block device support in Linux:

Past & Present

Definitions

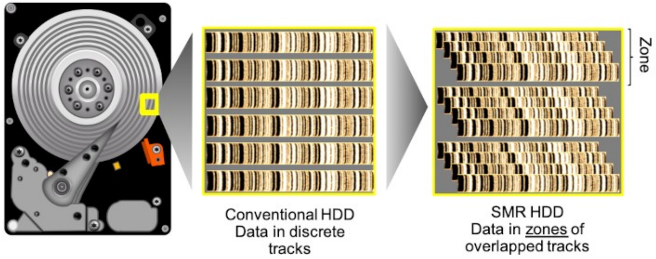
zone: A contiguous range of logical block addresses that are managed as a single unit.

zoned block device: A block device that consists of multiple zone

zone size: Size of a zone

chunk sectors (Linux): A device is divided up into chunks and merging is not allowed across two chunks. Required to be a **power-of-2 (po2)** value when **introduced**. **Zone size** is exposed as **chunk sectors** in block layer

SMR drives



SMR disks track organization

- Standards: ZAC/ZBC
- Overlapping tracks are grouped into bands called zones
- Zone size is always **a power-of-2**
- **Last zone** may have a **smaller** zone size (runt zone)

Image courtesy: zonedstorage.io

Zoned NAND flash

- Standards: NVMe ZNS, other standards
- A zone consists of multiple Erase Blocks(EBs)
- Usable LBAs in a zone is **not** a **power-of-2**

Definitions v2

zone: A contiguous range of logical block addresses that are managed as a single unit.

zoned block device: A block device that consists of multiple zone

zone size: Size of a zone. It needs to be **power-of-2** value to work in **Linux**

chunk sectors (Linux): A device is divided up into chunks and merging is not allowed across two chunks. Required to be a **power-of-2** value when **introduced**. Zone size is stored as chunk sectors in block layer

zone capacity: Usable logical blocks in a zone

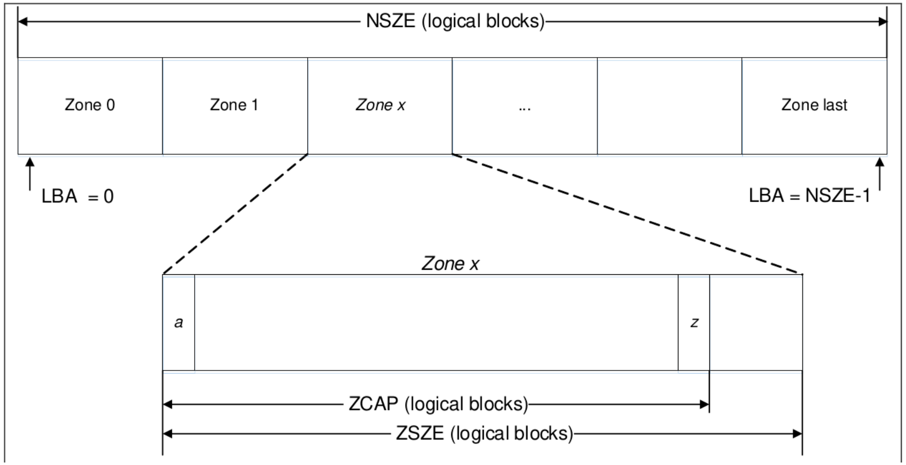
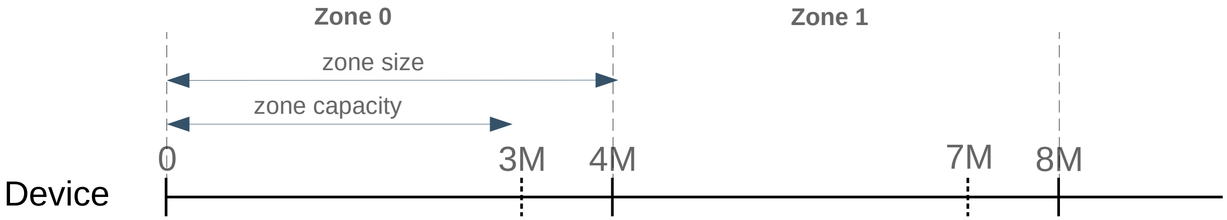


*Image taken from NVMe ZNS spec

LBA gaps

- **power-of-2(po2) zone size** is required to work in Linux leading to **LBA gaps** in ZNS devices
- **Reads** in LBA gaps behave like **deallocated blocks** (returns **zeroes** or a **pattern**)
- **Writes** in LBA gaps are **not allowed**
- SMR drives **do not** have this **problem**

A typical ZNS device layout with a po2 zone size :



zone capacity: 3M

zone size: 4M

Part 2

Non-power-of-2 zone size support in Linux

Why?

- Gaps between zone capacity and zone size for flash based zoned devices
  - Gaps inflate the LBA range above the usable size of a block device
  - Application needs to align to zone capacity and not zone size to reap the benefits
  - Gap introduces logic in the read path

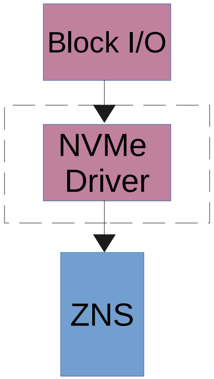
Why now?

- Linux removed the power-of-2 chunk sector constraint from v5.10[1]
- ZNS are being deployed in real environments
- New zoned standards are targeting NAND devices
  - Responsibility of the community is to ensure we support new tech without breaking backwards compatibility

[1] https://lore.kernel.org/all/20200911215338.44805-4-snitzer@redhat.com/

History

1) PO2 zone size device emulation in NVMe[1]:
   - No change needed in Userspace tools and filesystem
   - Complicates the NVMe driver
   - Cannot be reused by other drivers
   - LBA gaps



2) Add native support to block layer and filesystems[2]:
   - No LBA gaps
   - Breaks userspace tools for non-po2 zone sizes
   - BTRFS still in stabilization phase for zoned support, and superblock not power outage proof for non-po2 zone sizes

[1] https://lore.kernel.org/all/20220310094725.GA28499@lst.de/T/
[2] https://lore.kernel.org/all/20220516165416.171196-1-p.raghav@samsung.com/

GOAL

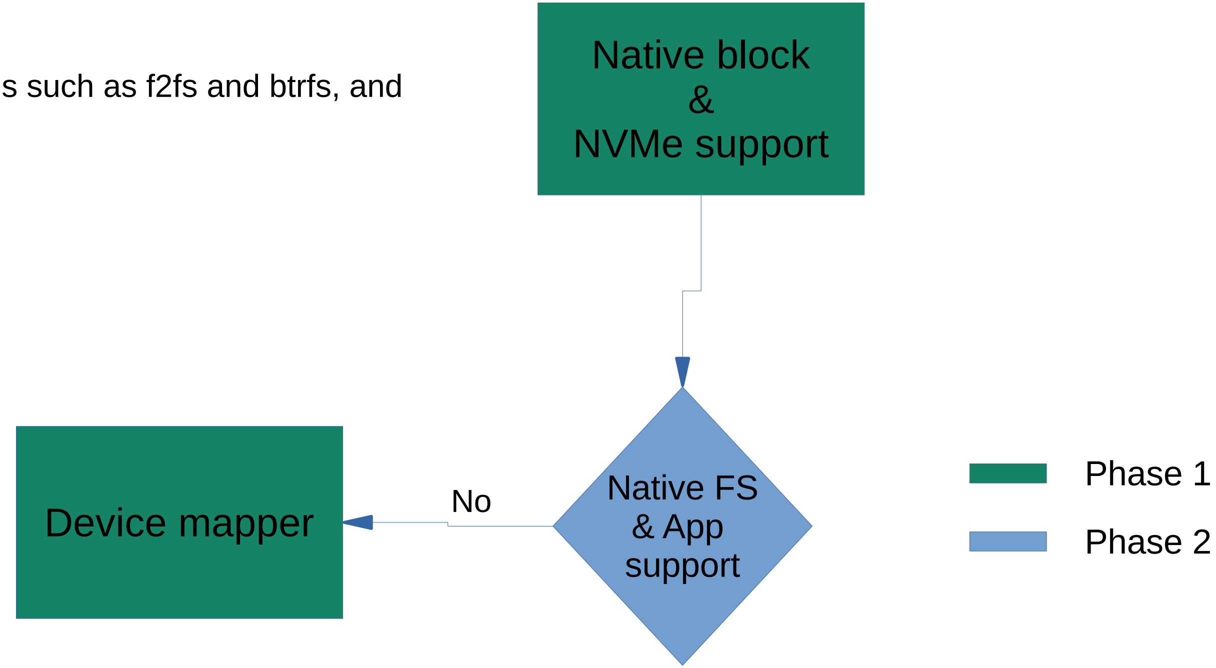
- Enable **non-power-of-2 zone sizes** in Linux for devices with **zone size == zone capacity**

- Ensure **compatibility** for non-power-of-2 zone size devices in existing filesystems and userspace applications **until native support** is added

Roadmap

Phase 1:
- Add native block layer and nvme driver support for non-po2 zone size drives
- Add a device mapper target to ensure compatibility of applications and filesystems for non-po2 zone size drives

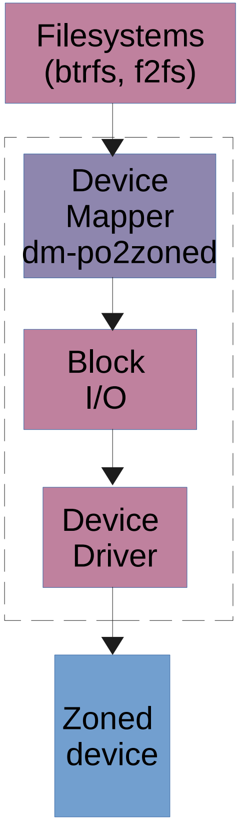
Phase 2:
- Add native non-po2 zone size drive support to filesystems such as f2fs and btrfs, and userspace applications



Current approach

- Native block layer and NVMe driver support for non-po2 zone sizes **without performance regression** for **po2 zone size devices**
  - No **LBA gaps** on raw block device for non-po2 zone size devices

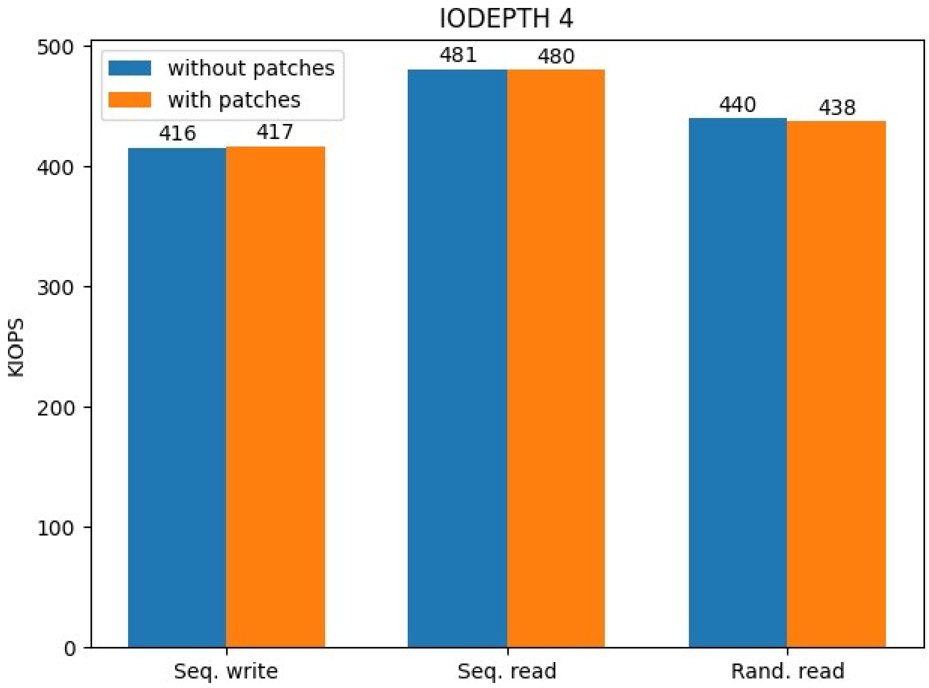
- Device mapper target **dm-po2zoned** to convert a **non-po2 zone size device** to a **po2 zone size target**
  - Avoids breaking userspace and filesystem for non-po2 zone size device **until native support is added**

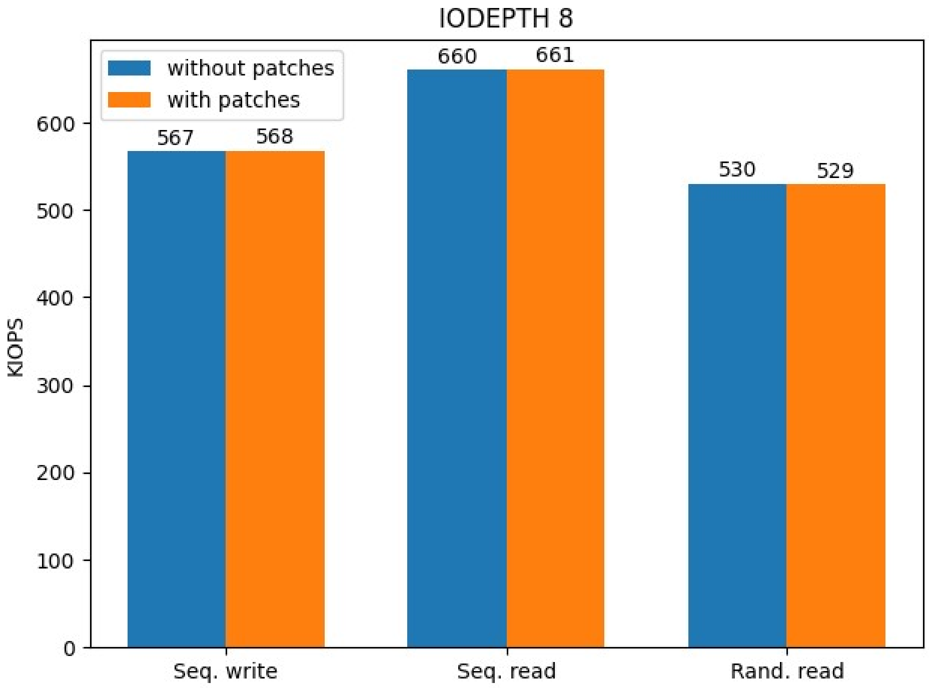


Native block layer support

Regression

No **performance regression** in **po2 zone size** devices and regular block device*



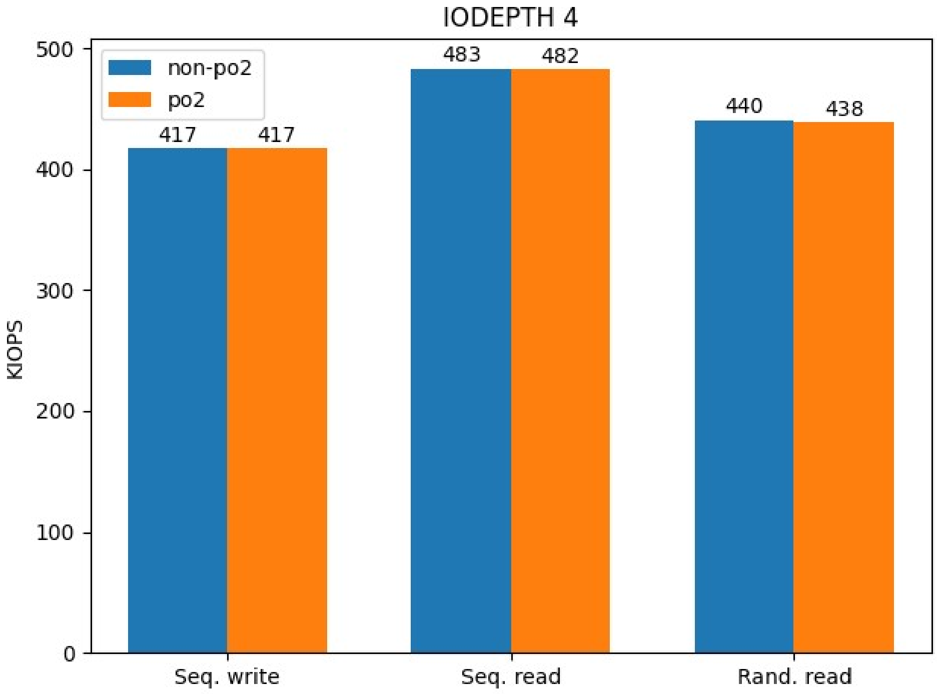


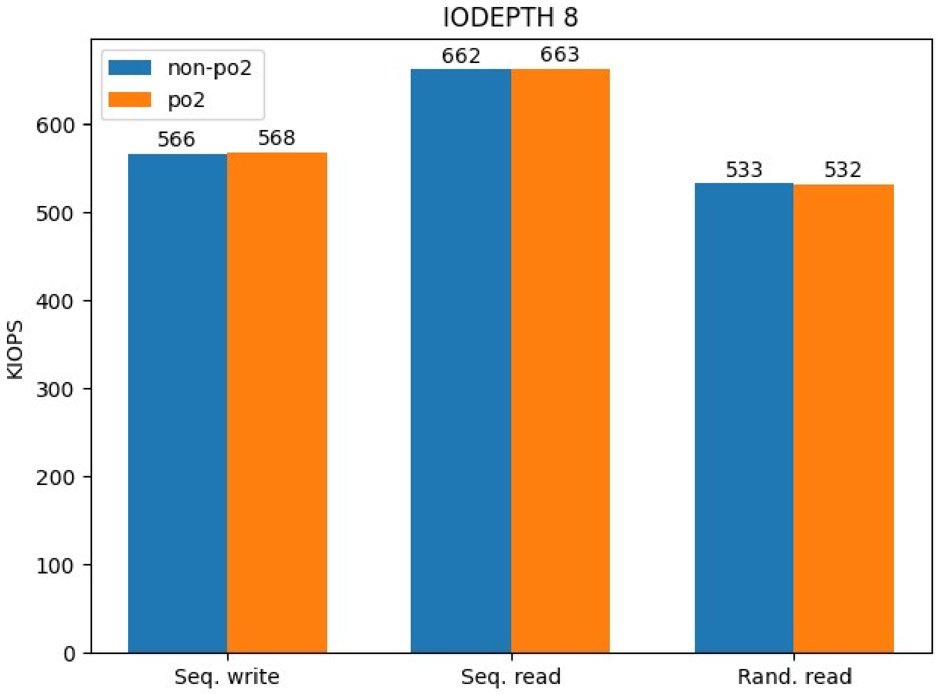
*Test conducted on a null blk with **128M** zone size device in a x86 box

Native block layer support

Progression

No **performance difference** between **po2 zone size** and **non-po2 zone size** devices*

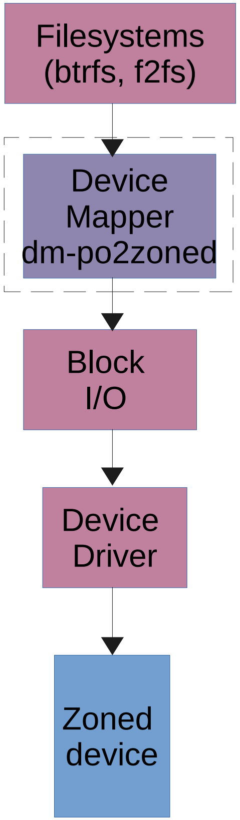


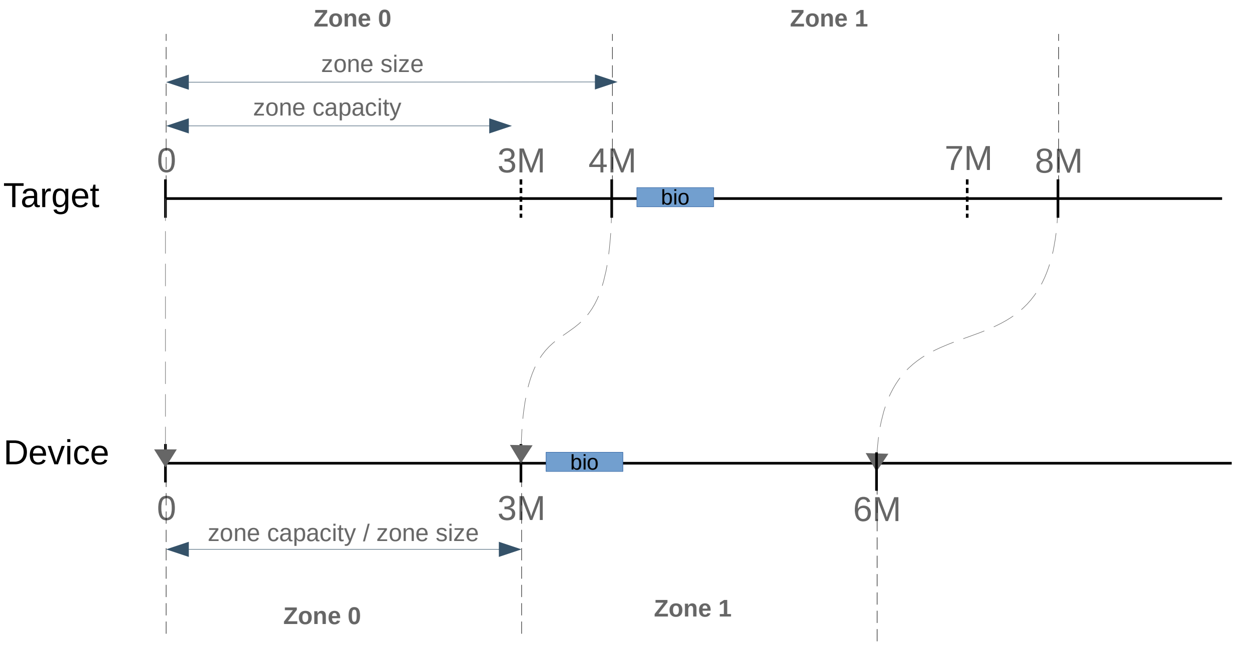


*Test conducted on a null blk with **128M** and **96M** zone size devices in a **x86 box**

dm-po2zoned

Algorithm

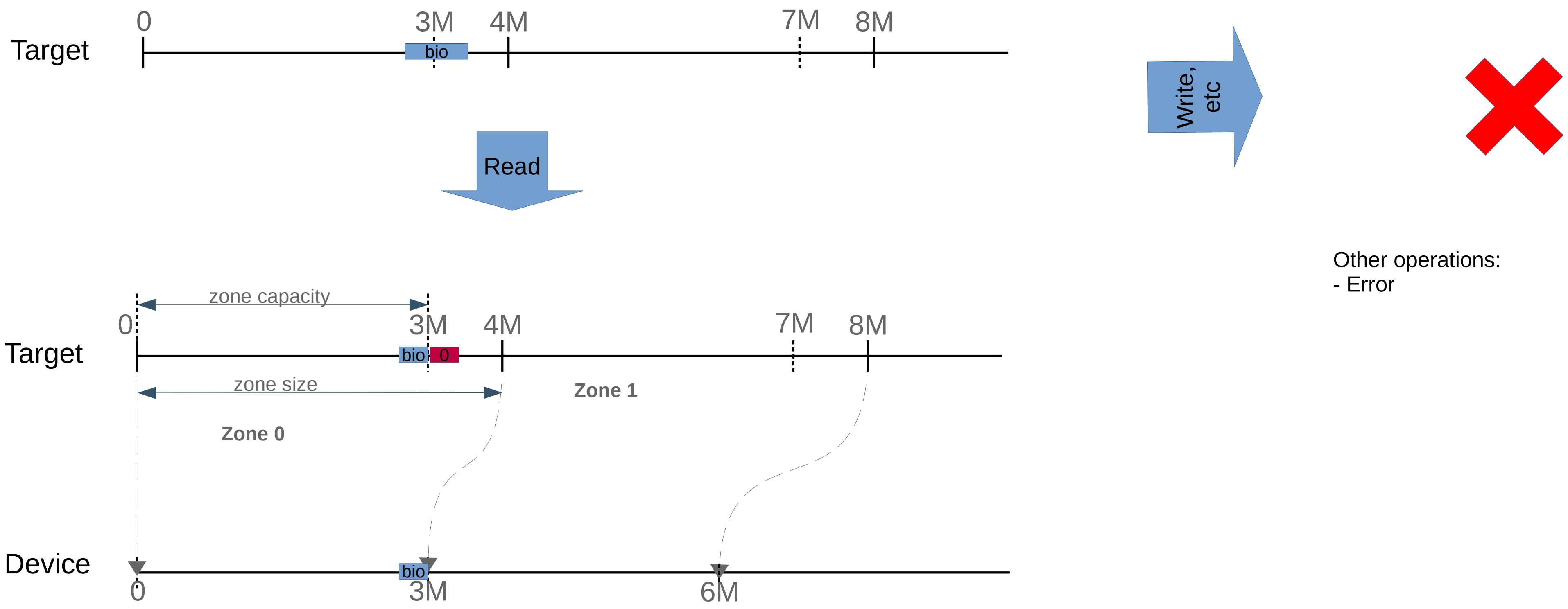




Device: 3M zone capacity and zone size
Target: 3M zone capacity and 4M zone size

dm-po2zoned

Handling bio in the emulated zone area

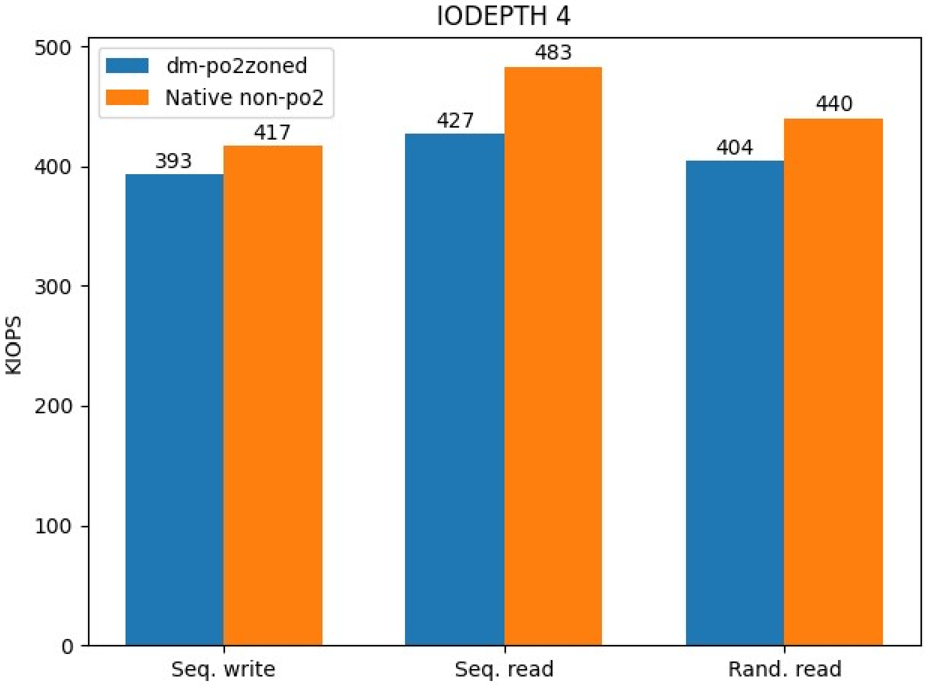


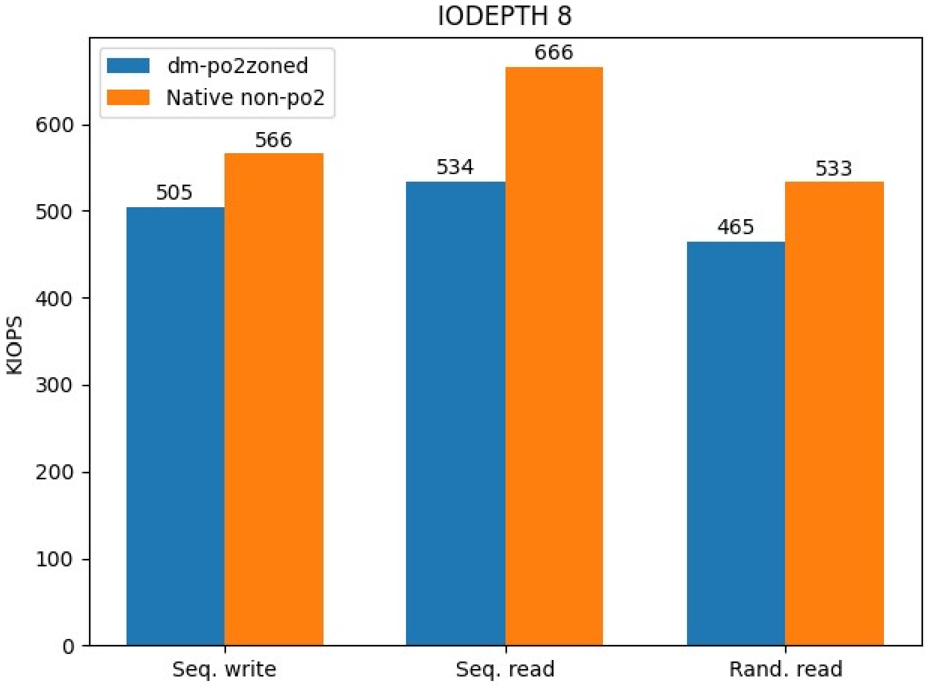
Other operations:
- Error

Read:
- split the bio across emulation boundary and fill zeroes on the split bio in the emulated area

dm-po2zoned

Device mapper cost: 8~15% average performance hit





*Test conducted on a null blk with 96M zone size device

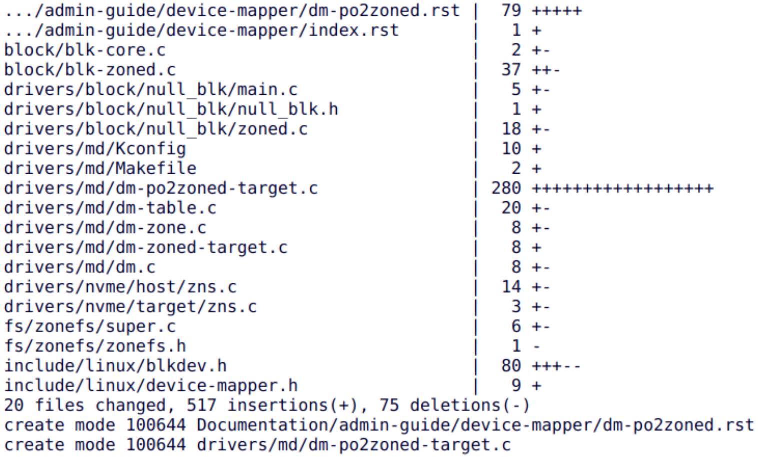
Conclusion

- Zoned block devices with non-po2 zone sizes can be safely supported in Linux
  - **No regression** in the **hot path** for **po2 zone size** devices
  - **Applications** can use **dm-po2zoned** to consume **non-po2 zone** size block devices **until native support** is added

Status & Future work

Status:
- Currently in v13 revision[1]

- Tested with blktest, zonefs test suite and fio



```
.../admin-guide/device-mapper/dm-po2zoned.rst |  79 +++++
.../admin-guide/device-mapper/index.rst       |   1 +
block/blk-core.c                              |   2 +-
block/blk-zoned.c                             |  37 ++-
drivers/block/null_blk/main.c                 |   5 +-
drivers/block/null_blk/null_blk.h             |   1 +
drivers/block/null_blk/zoned.c                |  18 +-
drivers/md/Kconfig                            |  10 +
drivers/md/Makefile                           |   2 +
drivers/md/dm-po2zoned-target.c               | 280 ++++++++++++++++++
drivers/md/dm-table.c                         |  20 +-
drivers/md/dm-zone.c                          |   8 +-
drivers/md/dm-zoned-target.c                  |   8 +
drivers/md/dm.c                               |   8 +-
drivers/nvme/host/zns.c                       |  14 +-
drivers/nvme/target/zns.c                     |   3 +-
fs/zonefs/super.c                             |   6 +-
fs/zonefs/zonefs.h                            |   1 -
include/linux/blkdev.h                        |  80 +++--
include/linux/device-mapper.h                 |   9 +
20 files changed, 517 insertions(+), 75 deletions(-)
create mode 100644 Documentation/admin-guide/device-mapper/dm-po2zoned.rst
create mode 100644 drivers/md/dm-po2zoned-target.c
```

Future work:

- Add native support to non-po2 zone sizes in filesystems(such as btrfs and f2fs) and userspace applications
- Add non-po2 zone size support in SCSI for Zoned UFS[2]

[1] https://lore.kernel.org/linux-block/20220912082204.51189-1-p.raghav@samsung.com/
[2] https://lore.kernel.org/lkml/024d16ac-d685-0fcf-1ad3-e99946852b03@acm.org/

Acknowledgments

Reviewers

- Damien Le Moal
- Hannes Reinecke
- Bart Van Assche
- Johannes Thumshirn
- Mike Snitzer

Questions?

Help by reviewing the patchset here[1] to make it a part of the next release

[1] https://lore.kernel.org/linux-block/20220912082204.51189-1-p.raghav@samsung.com/

Extras

Native block layer support

Regression

No **performance regression** in **po2 zone size** devices and regular block device*

IO DEPTH		4	8	DIFF
Seq. Write (KIOPS)	Without patches	416	567	±1%
	With patches	417	568	
Seq. read(KIOPS)	Without patches	481	660	
	With patches	480	661	
Random read(KIOPS)	Without patches	440	530	
	With patches	438	529	

*Test conducted on a null blk with **128M** zone size device in a x86 box

Native block layer support

Progression

No **performance difference** between **po2 zone size** and **non-po2 zone size** devices*

IO DEPTH		4	8	DIFF
Seq. Write (KIOPS)	non-po2	417	566	±1%
	po2	417	568	
Seq. read(KIOPS)	non-po2	483	662	
	po2	482	663	
Random read(KIOPS)	non-po2	440	533	
	po2	438	532	

*Test conducted on a null blk with **128M** and **96M** zone size devices in a **x86 box**

dm-po2zoned

Device mapper cost: 8~15% average performance hit

IO DEPTH		4	8	diff
Seq. Write (KIOPS)	Native non-po2	417	566	-8.2%
	dm-po2zone	393	505	
Seq. read(KIOPS)	Native non-po2	483	666	-15.65%
	dm-po2zone	427	534	
Random read(KIOPS)	Native non-po2	440	533	- 10.5%
	dm-po2zone	404	465	

*Test conducted on a null blk with 96M zone size device