Experiences implementing zonefs support in ZenFS

Jørgen Sværke Hansen <jorgen.hansen@wdc.com>
Western Digital

Linux Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022

Introduction

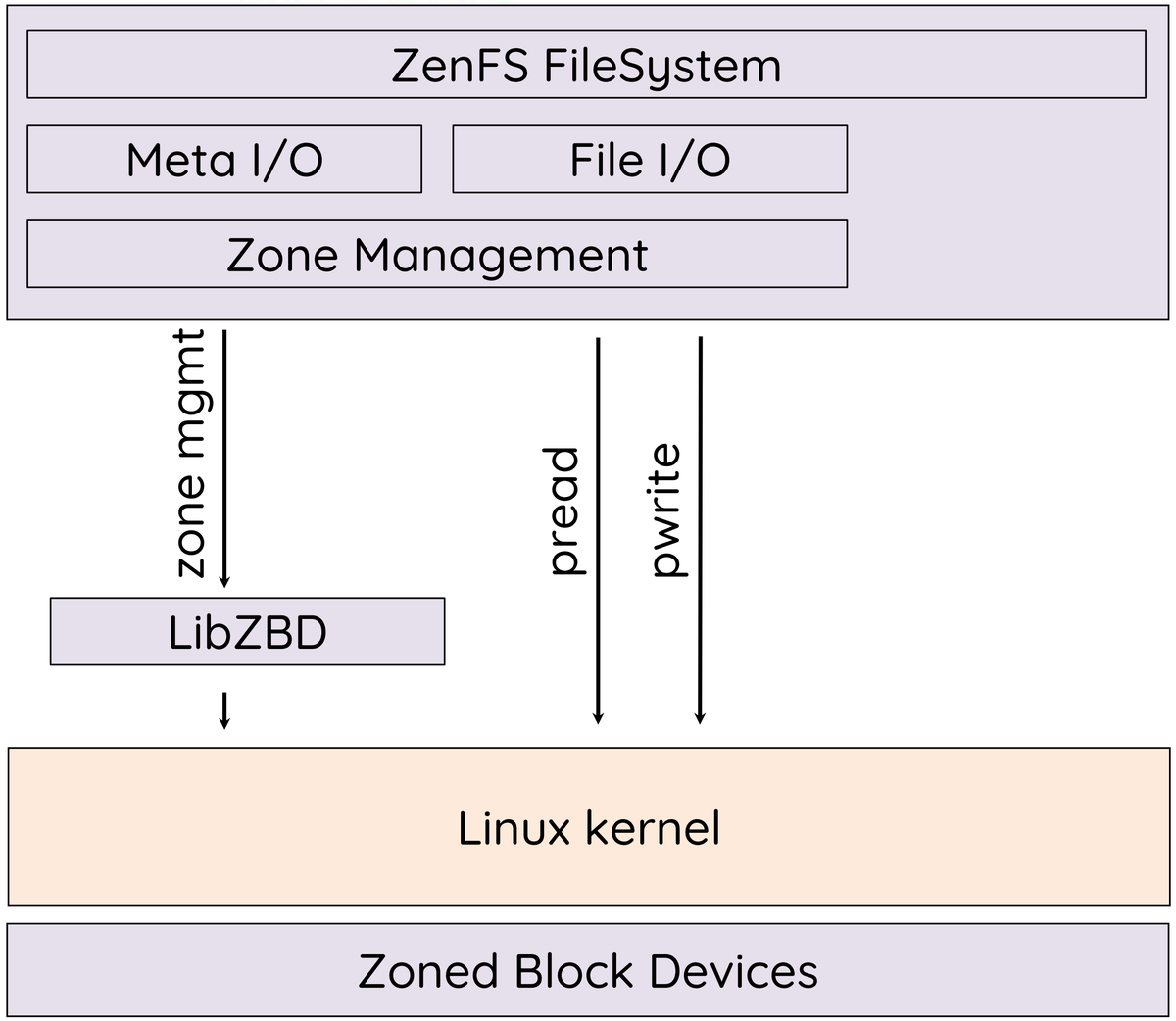
- ZenFS v1/v2 uses raw block device access through libzbd to access zoned storage

- Add zonefs support to ZenFS to:
  - Allow ZenFS users to take advantage of zonefs features such as permanent user permission settings
  - Allow ZenFS to run in containers or virtual machines using file system passthrough

- In the following:
  - What did it take to add support for zonefs
  - Are there any performance differences between zbdlib and zonefs

ZenFS 1.0/2.0 Architecture

Linux Plumbers Conference
Dublin, Ireland September 12-14, 2022

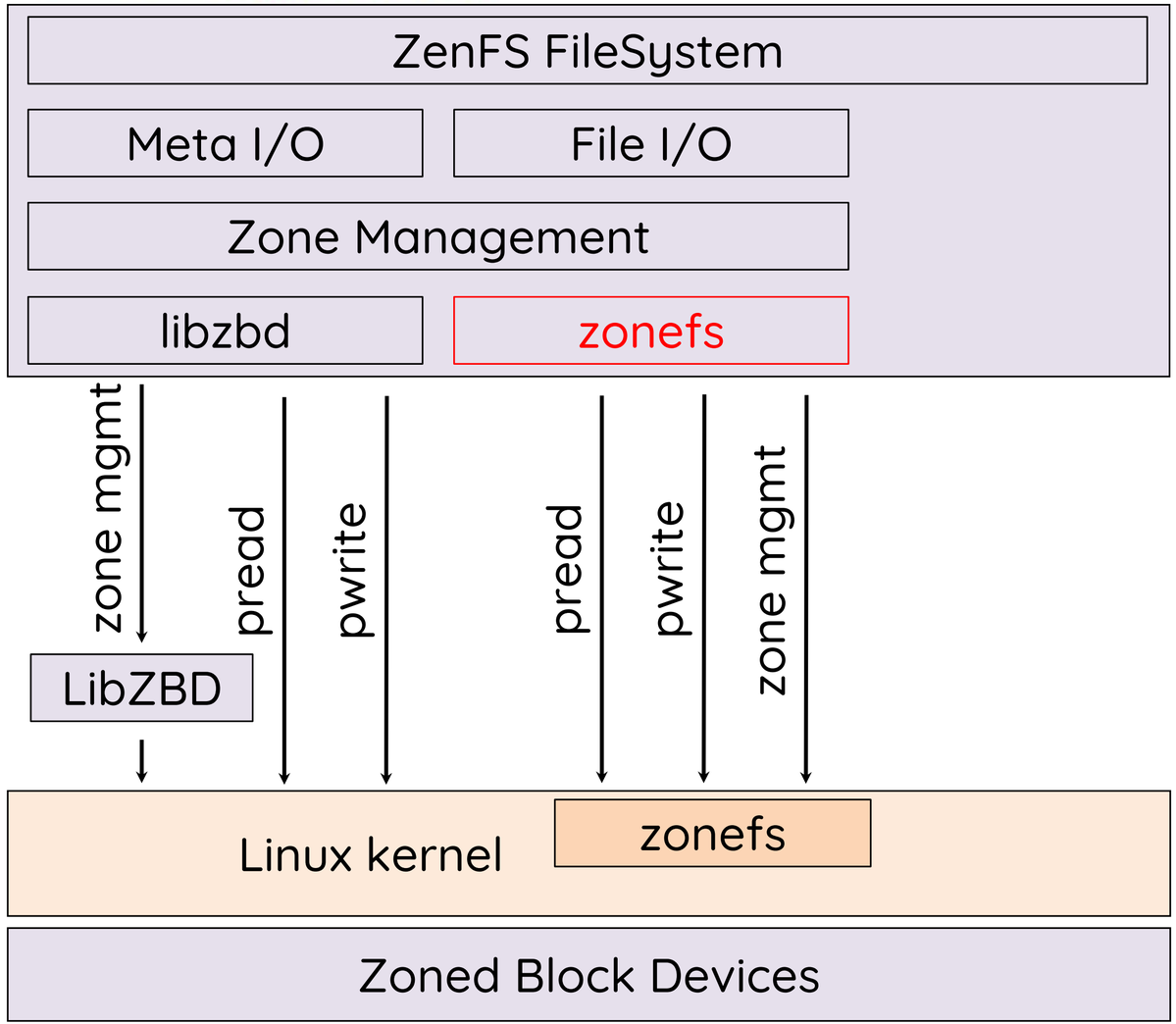


- Zone management using libZBD:
  - libZBD uses ioctl calls on the zoned block device node to do zone management, e.g.:
    - reset, finish, close
    - obtain limits on open/active zones
    - get zone size, capacity
  - Tracks open/active zones and closes/finishes writable zones to stay within device limits
- Read/write operations
  - Regular read/write operations directly on the zoned block device using LBAs

Source: https://github.com/westerndigitalcorporation/zenfs

Adapting ZenFS to use Zonefs

Linux Plumbers Conference
Dublin, Ireland September 12-14, 2022



- Zone management
  - Add new ZenFS URI for zonefs mount point: *zenfs://zonefs:<zonefs mountpoint>*
  - Refactor zone management to allow different zone block device backends
- Read/write operations
  - Upper layers assume a single LBA space
    - Convert single LBA space access into per zone file access
  - Management of open/active zones:
    - Mount zonefs with option explicit-open:
      - Open zones on the ZBD tied to open/close of writable zone files

Zone Management in ZenFS on Zonefs

Each zone is represented by a file and the zone operations are handled as follows:

- Reset:
  - Truncate to size 0
- Finish:
  - Truncate to zone capacity
- Close:
  - Close file
- Limits on open/active zones:
  - Obtained through procfs and sysfs (introduced in Linux v5.19)
- Zone count:
  - Obtained through fstat on directory
- Zone size, capacity:
  - Obtained through fstat on zone file

ZenFS File Operations on Zonefs

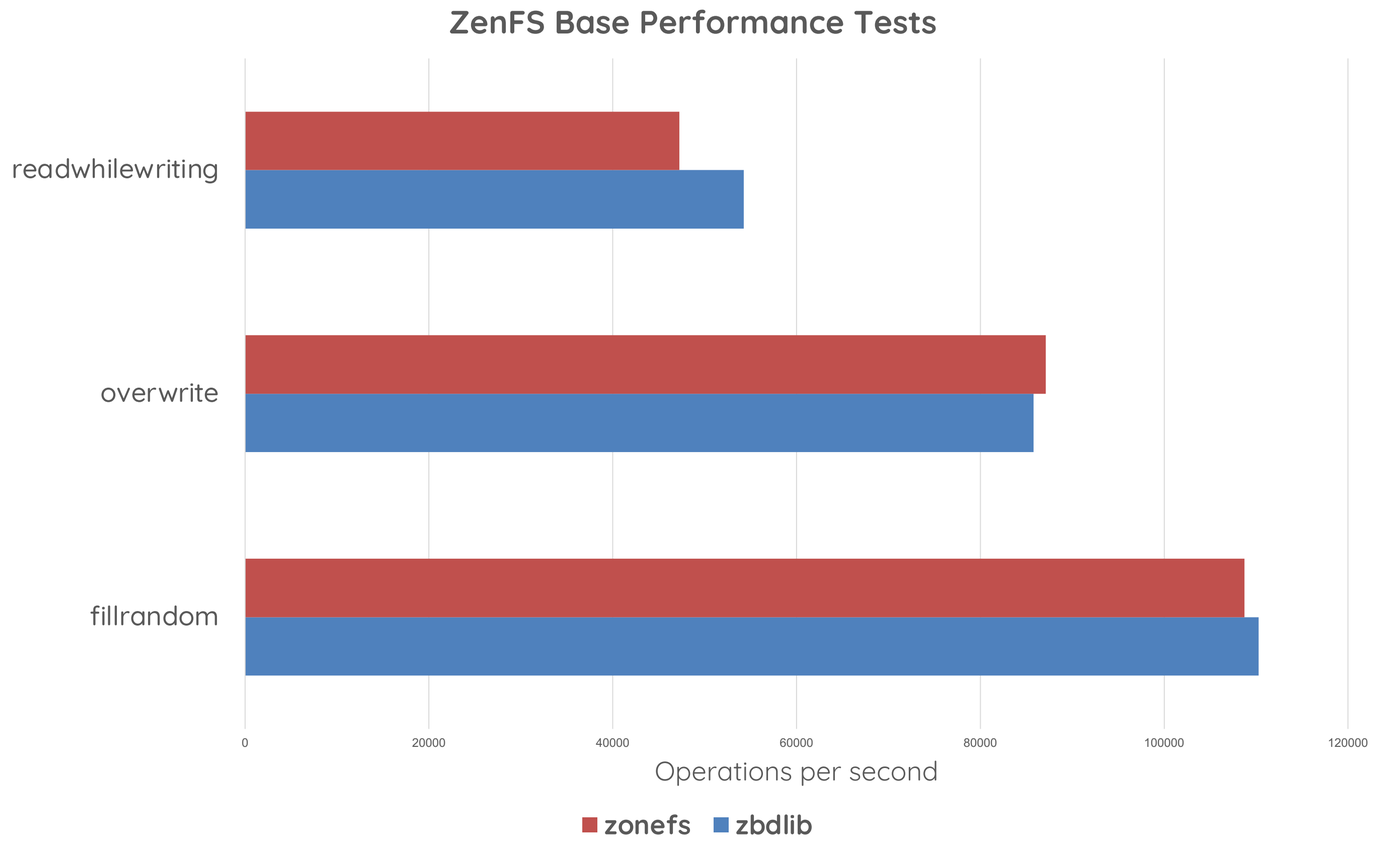
- Each zone is a file, so LBA based access is converted to <zone, byte offset>:
  - Read operation:
    - Open zone file(s) for reading (if necessary)
    - Keep FD in an LRU cache
  - Write operation:
    - Open zone file for writing (if necessary):
      - zonefs will open the zone as well (explicit-open mount option)
    - Cache FDs for zone files opened for writing until:
      - an explicit zone close from the upper ZenFS layers is received
      - a zone transitions to full or empty
    - Closing FD triggers zonefs to close zone on ZBD

Performance Comparison

- Comparing the performance of ZenFS using zbdlib and zonefs using db_bench with the base performance suite from ZenFS.
- Test setup:
  - Hardware: Single AMD Epyc 7313 16-core, 128GiB RAM
  - Kernel version: 5.19-rc4
  - NVME zoned block device:
    - Western Digital Ultrastar DC ZN540 (8TB)
    - Deadline scheduler enabled

Performance Comparison: Base Performance

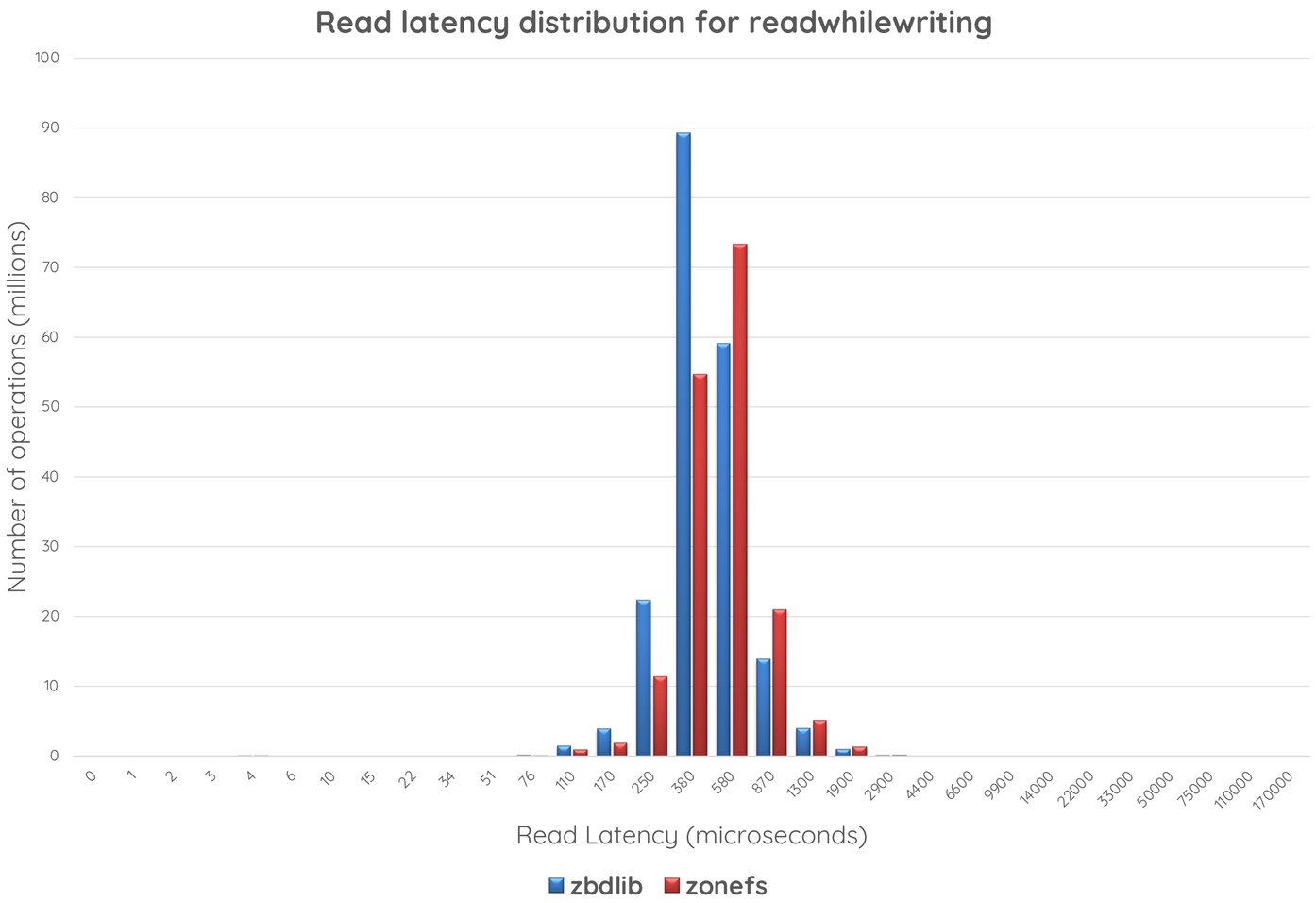
Linux Plumbers Conference
Dublin, Ireland September 12-14, 2022



- Zonefs and zbdlib performance are close:
  - readwhilewriting shows the largest difference:
    - Zonefs: 47.2K ops/s
    - Zbdlib: 54.3K ops/s

Linux Plumbers Conference
Dublin, Ireland September 12-14, 2022

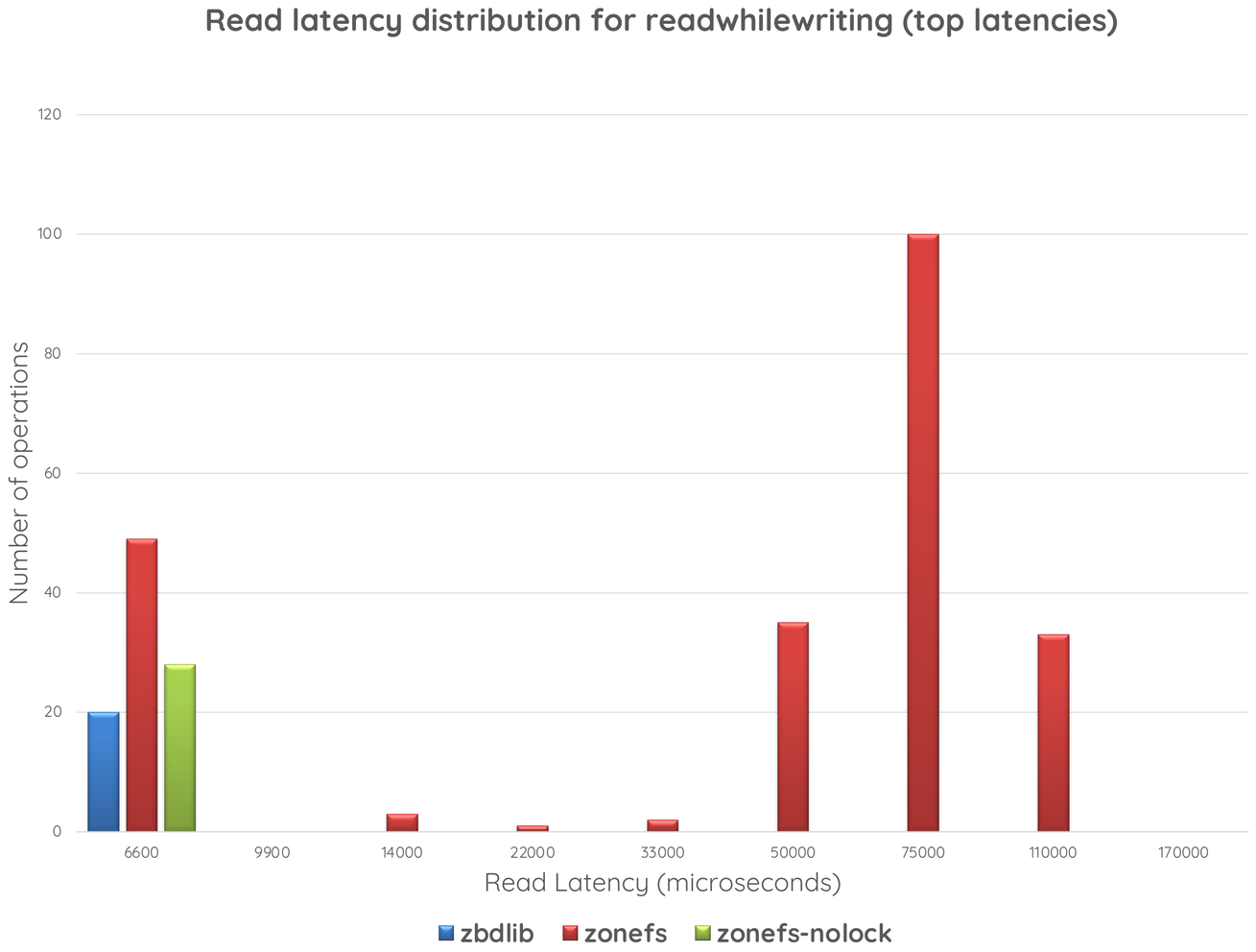
Performance: Read Latency Distribution



- Average latencies slightly higher for zonefs
- A small number (<200) of operations have very high latencies (55+ milliseconds)

Focus on Highest Latencies

Linux Plumbers Conference
Dublin, Ireland September 12-14, 2022



A small number of operations have very long latencies!

Profiling revealed that read operations where blocked while zones are being finished:

- A finish operation is a truncate to size of zone capacity
- File system semantics block read/write operations while a truncate is in progress
- For a ZNS SSD, a finish can take hundredths of microseconds
- The ZNS SSD itself allows reads while a zone is being finished, so libdZBD implementation doesn't experience this issue
- Removing read locks confirms this (zonefs-nolock in graph)

Future work: determine if read lock during truncate can be relaxed

Conclusion

- Straight forward to adapt existing existing zbdlib based code to also support zonefs
- Performance is roughly the same for zonefs and zbdlib, although zonefs may see higher latencies for operations happening concurrelty with a zone finish
- Code is upstreamed and available at:
  https://github.com/westerndigitalcorporation/zenfs