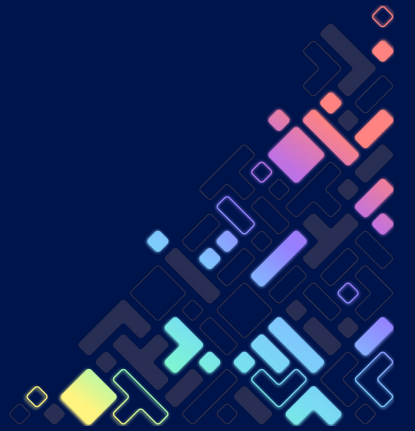
LINUXCON@ OPEN SOURCE SUMMIT NORTH AMERICA

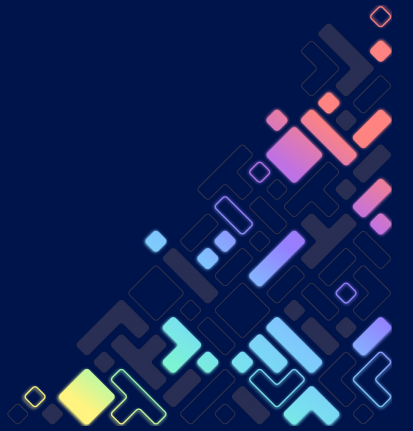
THE LINUX FOUNDATION

EROFS: Past, Present, and Future

Xiang Gao / Tianyi (Cardy) Tang
Alibaba Cloud



Introduction

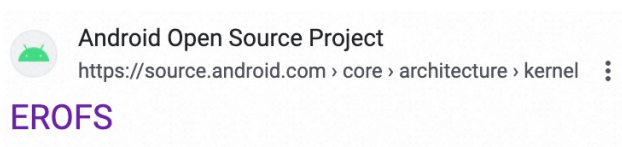


What's EROFS?

- Enhanced Read-Only File System (started from late 2017)
  - Formally available since Linux kernel 5.4 (and Fedora 30, Debian 11, Ubuntu 20.04, …).
- A modern, flexible, **high-performance** block-based in-kernel immutable filesystem
  - Block-based, NOT quite blockdevice-based;
  - On-disk data is strictly block-aligned. No extra data movement & I/O amplification + DAX + Direct I/O;
  - Although it's simple, but more than just an archive format. We leverage in-kernel possibilities from kernel developer perspective as much as possible.
- A minimal core EROFS has:
  - Unencoded data;
  - Built-in UUID & volume label;
  - Direct I/O;
  - (Optional) Block-based chunk deduplication;
  - (Optional) Optimized Extended Attributes and POSIX ACL;
  - (Optional) Built-in layering support;
  - (Optional) FSDAX support to share cache memory between hosts and guests;
- Applications

What's EROFS?

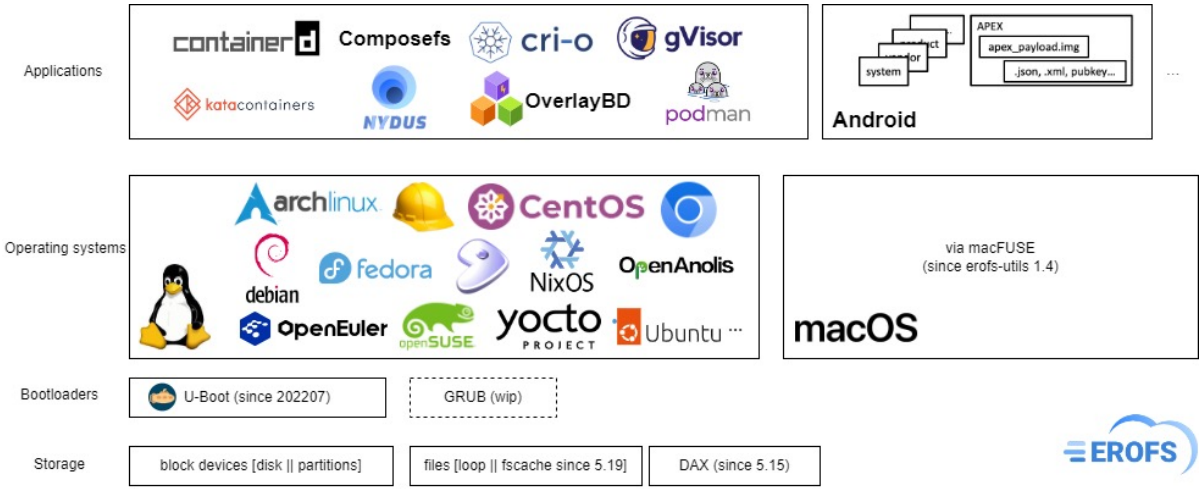
- A full-featured compressed EROFS supports:
  - (Optionally) LZ4, LZMA, DEFLATE transparent compression on a per-inode basis;
  - (Optionally) Rolling-hash compressed data deduplication;
  - (Optionally) Tail-end compressed data deduplication (fragments);
  - (Optionally) Tail-end compressed data inlining (ztailpacking);
- Typical applications



Linglong

EROFS Community

- Maintainers / reviewers:
  - Xiang Gao (Alibaba Cloud);
  - Chao Yu (OPPO);
  - Yue Hu (Coolpad Group);
  - Jingbo Xu (Alibaba Cloud);
  - Sandeep Dhavale (Google);
  - (add you here…)



EROFS documentation site: https://erofs.docs.kernel.org

Contributed vendors [1]

- Alibaba Group
- Bytedance
- Coolpad
- Google
- Huawei
- OPPO
- Red Hat
- Shanghai Jiao Tong University
- …

[1] https://erofs.docs.kernel.org/en/latest/credits.html

The history

- Android has several read-only partitions which behave as system fireware, which means “Android core can only be changed by way of an update”
- Firmwares become larger and larger, so more free space is needed on brand new Android devices, especially on low-ended devices.
- Why introducing EROFS? [1]
  - Dynamic runtime performance especially **in low memory scenarios with constant heavy memory pressure** on Android.
  - All previous ones don’t work well to meet our performance needs as competitive products on market.
  - Our goal: simple but effective to cover various immutable use cases from kernel filesystem developer perspective.

[1] https://www.usenix.org/conference/atc19/presentation/gao

Why EROFS?

Security & simplicity

- Reduced on-disk metadata complexity;
- Read/Write Splitting;
- Avoiding unnecessary codebase coupling;
- Data tampering prevention.

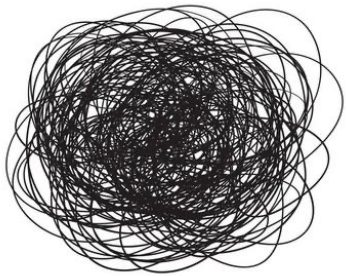
Faster build performance;

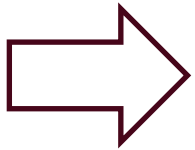
Smaller overall filesystem sizes;

Easier to merge filesystems from sub-images;

Less intricate bugs;

Smaller attack surface;





Especially on runC, yet in generally it's preferred to reuse the same images for runC and/or VM-based runtimes.

Why EROFS?

ext4 subsystem

List(s): linux-ext4@vger.kernel.org
Fixed bugs: 134
Parent subsystem(s): fs (103)

open (27):

Title	Repro	Cause bisect
WARNING: locking bug in ext4_xattr_inode_update_ref (3) ...	C	
KASAN: slab-use-after-free Read in fsnotify ext4	C	
WARNING in ext4_xattr_set_entry (2) ext4	C	
possible deadlock in ext4_xattr_inode_iget (3) ext4	C	error
WARNING: locking bug in ext4_xattr_inode_iget (2) ext4	C	
possible deadlock in __ext4_mark_inode_dirty (3) ext4		
possible deadlock in ext4_evict_inode (3) ext4		

syzbot issues may be derived from:

- Crafted malicious images;
- Less common compatible or redundant (for immutable image use cases) on-disk formats, increasing code coupling;
- Combined with the above, extra twisted write paths (+journaling) challenge the overall stability again.

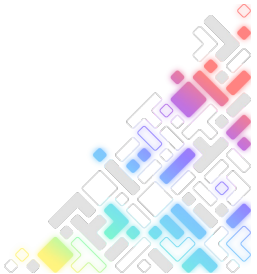
POC: a malicious EXT4 filesystem image which can panic the kernel (on CentOS stream 9)

```
[root@iZbp1cs6ivgi0jfla609xlZ ~]# uname -a
Linux iZbp1cs6ivgi0jfla609xlZ 5.14.0-354.el9.x86_64 #1 SMP PREEMPT_DYNAMIC Thu Aug 10 21:06:12 UTC 2023 x86_64 x86_64
x86_64 GNU/Linux
[root@iZbp1cs6ivgi0jfla609xlZ ~]# mount -o ro poc.ext4 mnt
[root@iZbp1cs6ivgi0jfla609xlZ ~]# dmesg | tail -n2
[   22.472478] EXT4-fs (loop0): write access unavailable, skipping orphan cleanup
[   22.472482] EXT4-fs (loop0): mounted filesystem without journal. Quota mode: none.
[root@iZbp1cs6ivgi0jfla609xlZ ~]# ls -l mnt
client_loop: send disconnect: Broken pipe
```

Why EROFS?

Flexibility

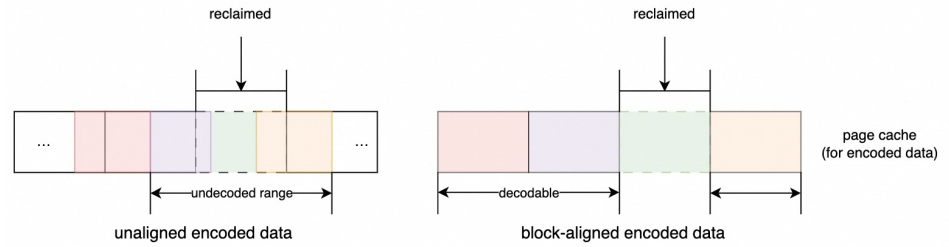
- In-kernel filesystem driver and/or user-space implementation;
- Support both block-device and file-based distributions;
- Per-file compression configuration: (un)compressed, different algorithms;
- Filesystem size dynamic resizing —— Just append meta(data) directly;
- On-demand on-disk format (metadata) enabling;
- Un-fixed on-disk layout arrangement
    - As long as the metadata can be parsed properly.



Why EROFS?

Performance [1]

- Due to block-aligned data, all data I/Os can be fully utilized as well as no memory movement and I/O amplification;
- Fixed-sized output compression for LZ4, LZMA and DEFLATE



[1] https://erofs.docs.kernel.org/en/latest/design.html

Why EROFS?

Performance [1]

- Test setup
  - Processor: x86_64, Intel(R) Xeon(R) Platinum 8369B CPU @ 2.70GHz * *2 Vcores*
  - *Storage: Cloud disk, 3000 IOPS upper limit*
  - *OS Kernel: Linux 6.2*
  - *Software: LZ4 1.9.3, erofs-utils 1.6, squashfs-tools 4.5.1*

[1] https://git.kernel.org/pub/scm/linux/kernel/git/xiang/erofs-utils.git/tree/docs/PERFORMANCE.md

Why EROFS?

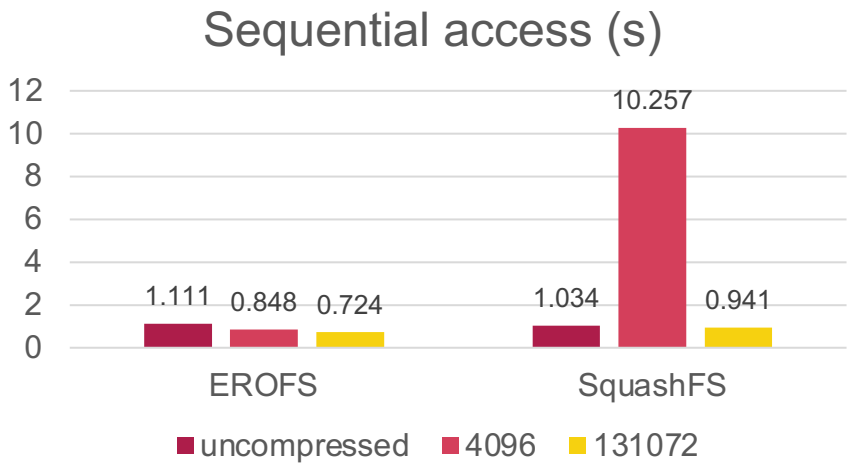
Multi-file performance (Debian docker image)

- Image sizes

Size	Filesystem	Cluster size	Build options
124669952	EROFS	uncompressed	-T0
124522496	SquashFS	uncompressed	-noD -noI -noX -noF -no-xattrs -all-time 0 -no-duplicates
73601024	SquashFS	4096	-b 4096 -comp lz4 -Xhc -no-xattrs -all-time 0
73121792	EROFS	4096	-zlz4hc,12 -C4096 -Efragments -T0
67162112	SquashFS	16384	-b 16384 -comp lz4 -Xhc -no-xattrs -all-time 0
65478656	EROFS	16384	-zlz4hc,12 -C16384 -Efragments -T0
59150336	SquashFS	131072	-b 131072 -comp lz4 -Xhc -no-xattrs -all-time 0
58515456	EROFS	131072	-zlz4hc,12 -C131072 -Efragments -T0

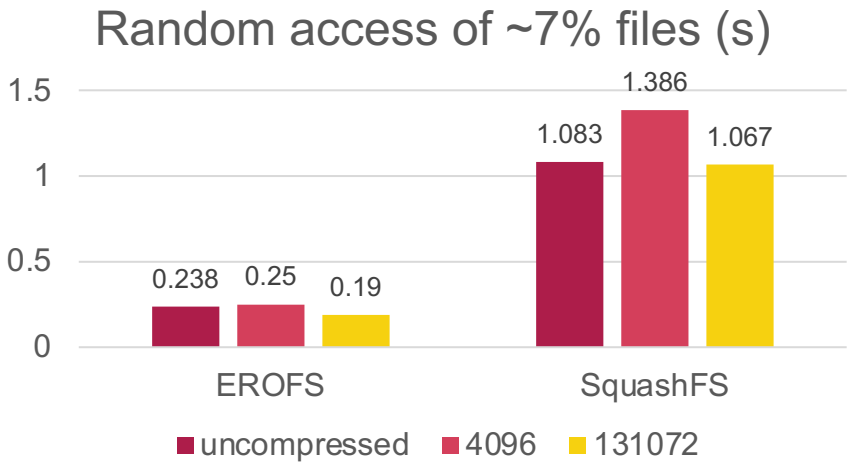
Why EROFS?

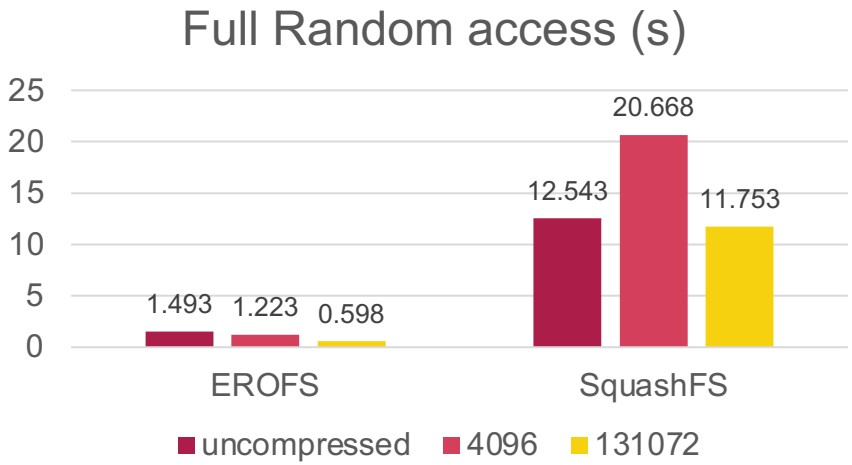
Multi-file performance (Debian docker image)



Why EROFS?

Multi-file performance (Debian docker image)





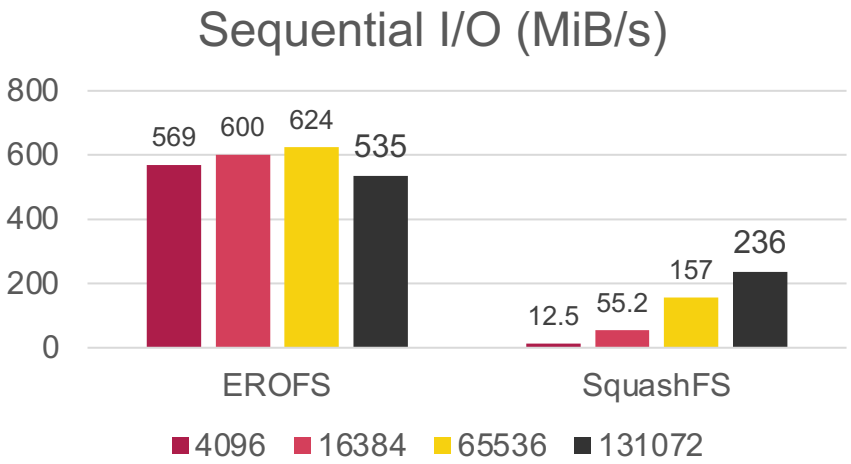
Why EROFS?

Single large file performance (silesia.tar, 203M)

Size	Filesystem	Cluster size	Build options
114339840	Squashfs	4096	-b 4096 -comp lz4 -Xhc -no-xattrs
104972288	EROFS	4096	-zlz4hc,12 -C4096
98033664	SquashFS	16384	-b 16384 -comp lz4 -Xhc -no-xattrs
89571328	EROFS	16384	-zlz4hc,12 -C16384
85143552	SquashFS	65536	-b 65536 -comp lz4 -Xhc -no-xattrs
81211392	SquashFS	131072	-b 131072 -comp lz4 -Xhc -no-xattrs
80519168	EROFS	65536	-zlz4hc,12 -C65536
78888960	EROFS	131072	-zlz4hc,12 -C131072

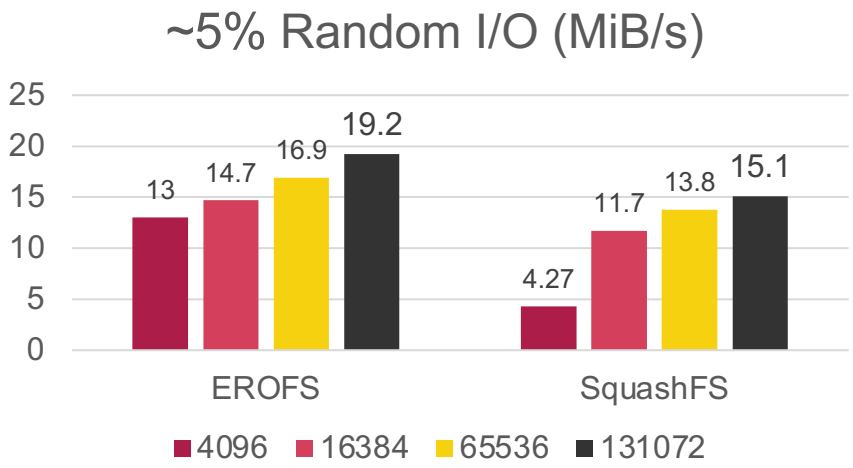
Why EROFS?

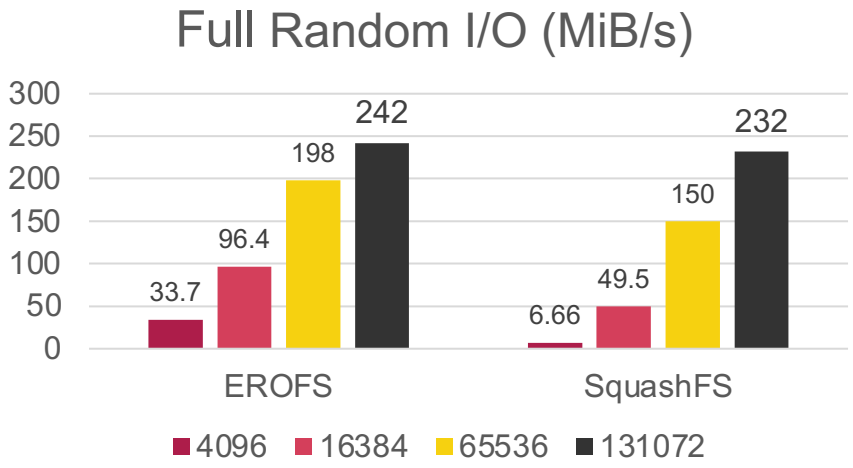
Single large file performance (silesia.tar, 203M)



Why EROFS?

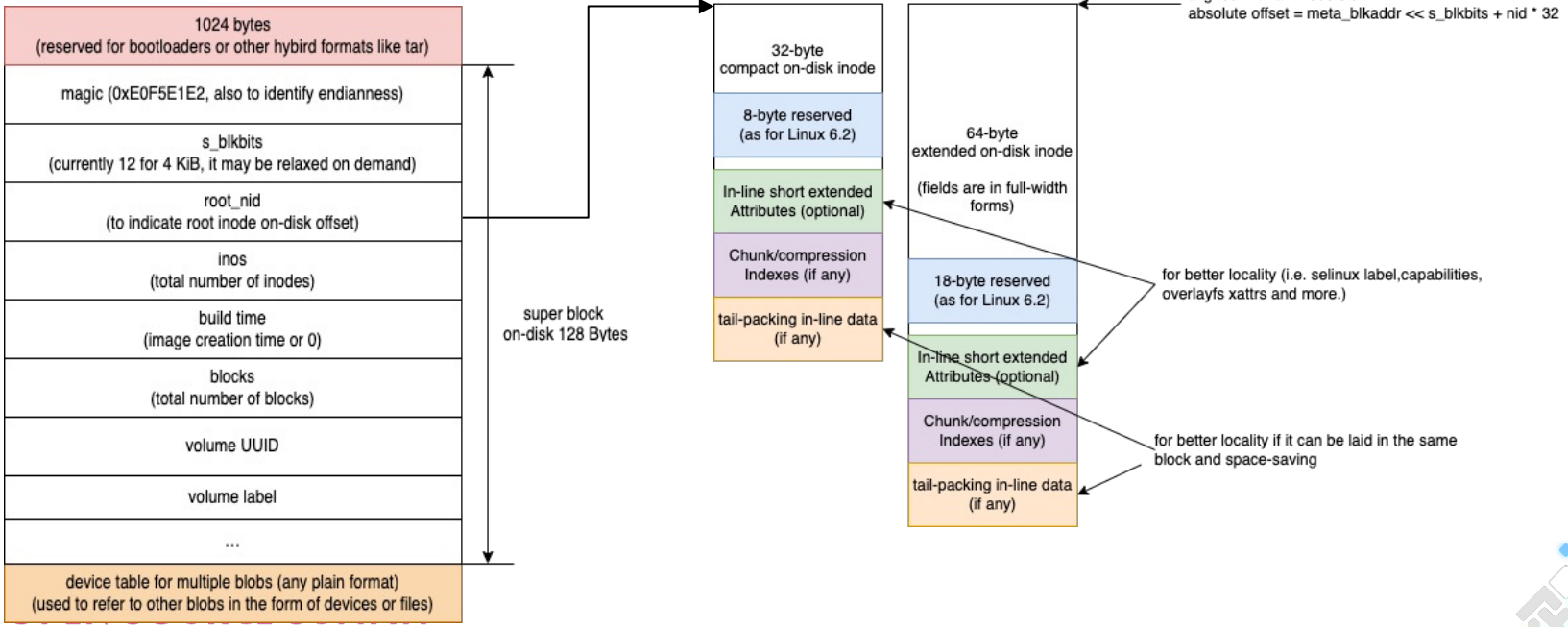
Single large file performance (silesia.tar, 203M)





EROFS core on-disk brief introduction

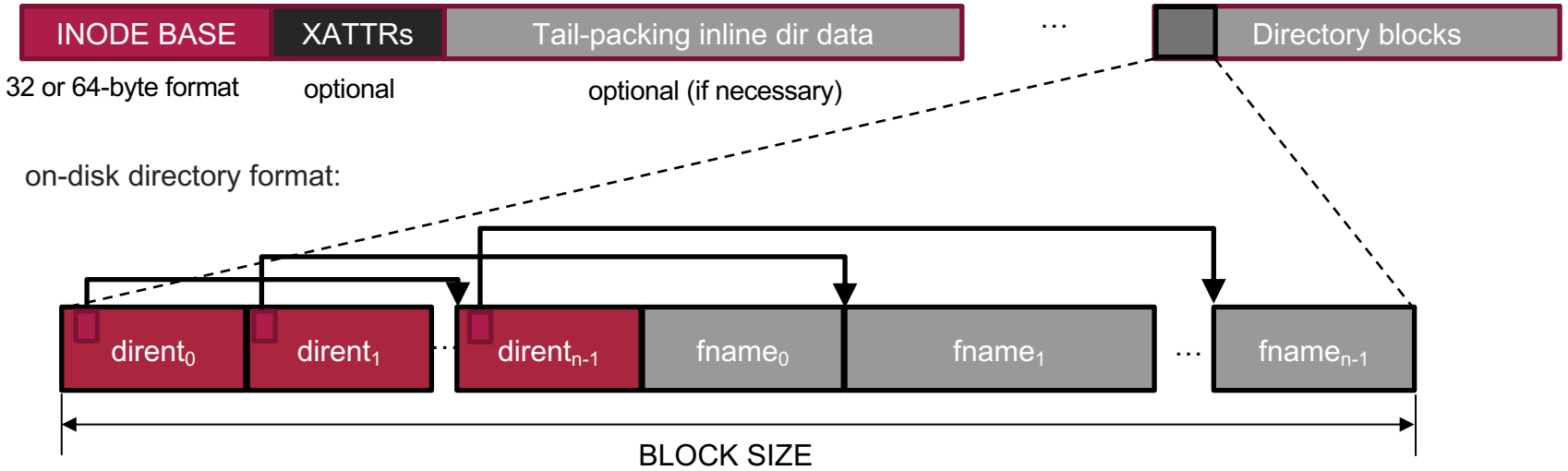
- Almost all EROFS on-disk structures are well-aligned and laid within a single filesystem block (never across two blocks for performance)
- On-disk super block as well as two version inodes (32 and 64 bytes)



EROFS core on-disk brief introduction

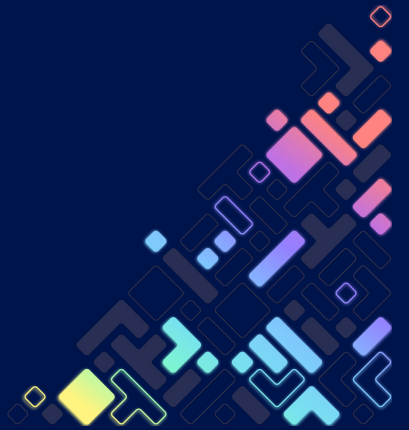
- **On-disk directory format**

Directory files:



Filenames sorted in alphabetical order to improve performance by binary search.
It also makes readdir() return in order (not necessarily though).

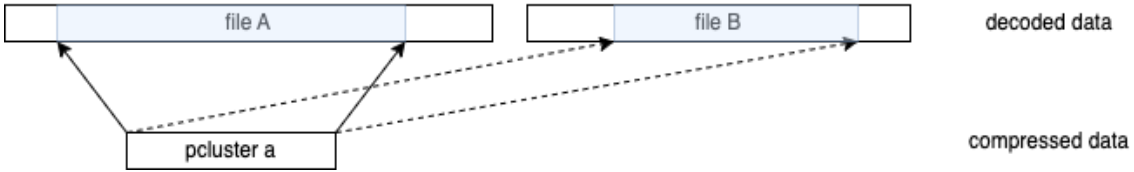
Recent Features



EROFS global (un)compressed data deduplication

- **Chunk-based data deduplication**
  - Uncompressed data

- **Global compressed data deduplication**
  - a pcluster can be referenced for multiple times in the prefix form;
  - byte-granularity deduplication with minimized extent length of 4KiB;
  - friendly to text files with small compressed pclusters;



enwiki 20230201 + 20230220 (first 100 pages, 20-day difference, 1.8G)	sub-file dedupe support	size
squashfs [4k]	⭕	1235M
erofs [4k, default]	⭕	1201M
squashfs [8k]	⭕	1150M
erofs [4k]	✅	1147M
squashfs [128k, default]	⭕	1110M
erofs [64k]	✅	988M

```
Dataset:                linux 5.10 + 5.10.50 + 5.10.100
Compression algorithm: lz4hc,12

Additional options:    -T0 --force-uid=1000 --force-gid=1000
              (in order to force 32-byte inode to match squashfs)

      4k  pcluster + fragment + dedupe  379 MiB
      8k  pcluster + fragment + dedupe  347 MiB
     16k  pcluster + fragment + dedupe  326 MiB
     32k  pcluster + fragment + dedupe  313 MiB
     64k  pcluster + fragment + dedupe  310 MiB

squashfs-tools 4.5.1 test results (which uses level 12 by default
for lz4hc):
     16k  block + fragment              409 MiB
     32k  block + fragment              365 MiB
     64k  block + fragment              334 MiB
    128k  block + fragment              312 MiB
```

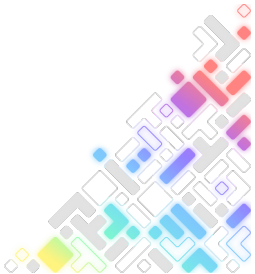
Case 1: Shared disk among nodes

Observations

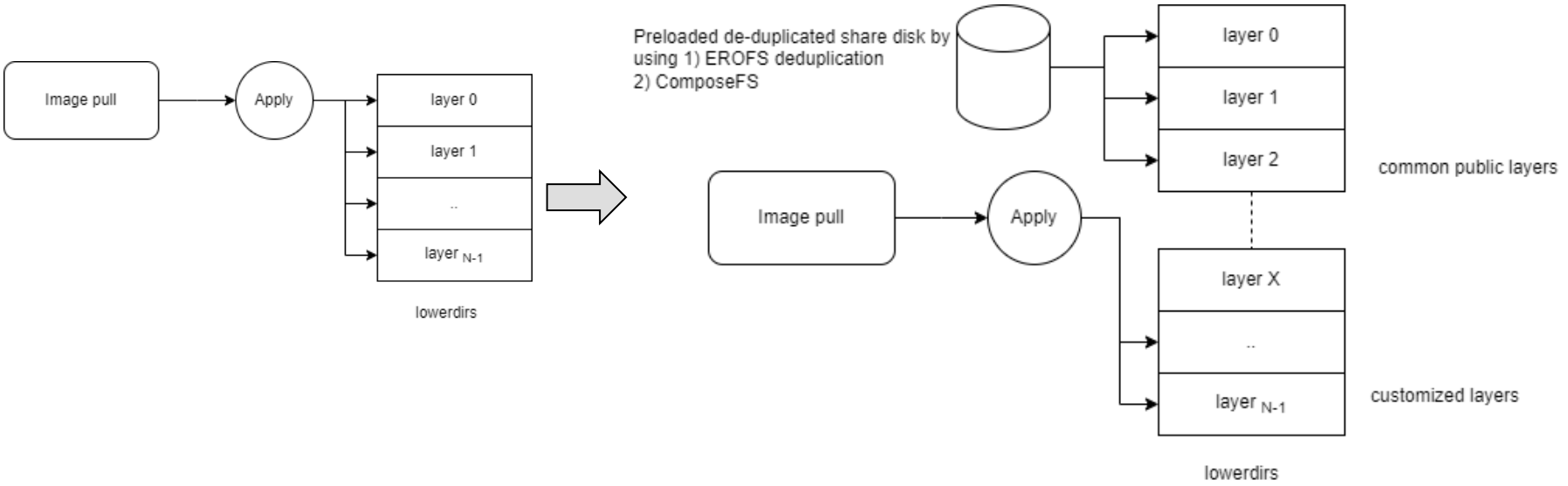
- Pulling all needed layers for each container image startup is slow;
- Most underlay base layers can be publicly available (e.g. installing package, etc.);
- The difference between minor security versions are quite small;

Ideas

- Introduce “shared disk (image)” to store most common base layers;
- Use EROFS (un)compressed deduplication to minimize images and I/Os.



Case 1: Shared disk among nodes



Case 1: Shared disk among nodes

- Deduplication between slightly different minor versions:
  - downloaded 10 versions of ubuntu:jammy from Docker Hub, specifically jammy-{20221130,20230126,20230301,20230308,20230425,20230522,20230605,20230624,20230804,20230816}

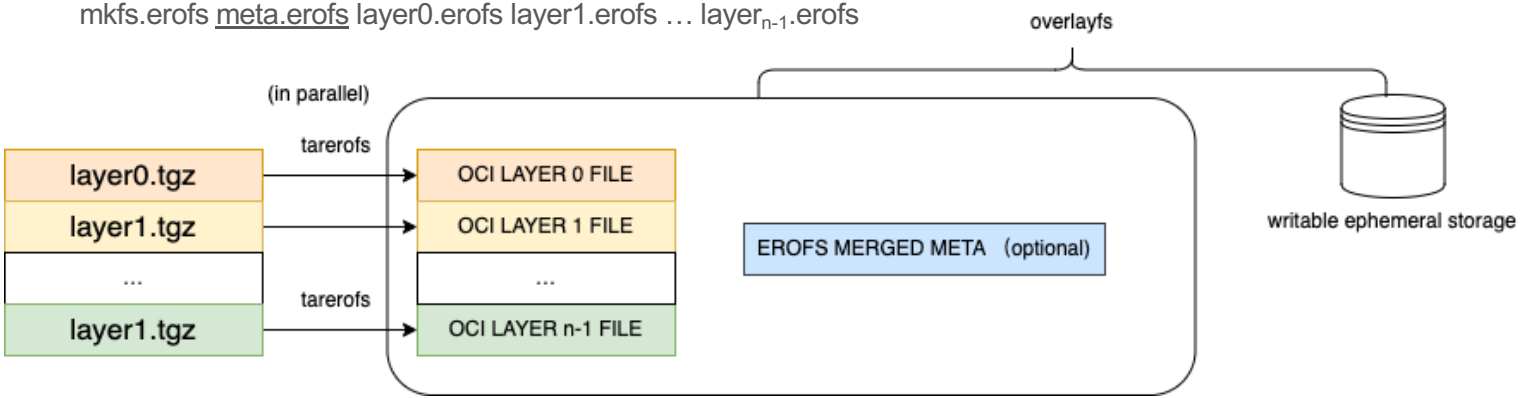
	Total Size (MiB)	Average layer size (MiB)	Saved / 766.1MiB
Compressed OCI (tar.gz)	282.5	28.3	63.1%
Uncompressed OCI (tar)	766.1	76.6	0%
Uncompressed EROFS (4k deduped)	109.5	10.95	86%
Compressed EROFS (DEFLATE,9,32k)	46.5	4.7	94%
Compressed EROFS (LZ4HC,12,64k)	53.8	5.4	93%
SquashFS with fragments (GZIP, 9,128k, -nol[1])	47.0	4.7	94%
SquashFS with fragments (LZ4HC,12,128k, -nol)	54.7	5.5	93%

[1] SquashFS uses –b 131072 by default, -nol will disable its metadata compression.

EROFS Native layering (sub-filesystem merging)

- sub-images can be merged as multiple layers with a tiny metadata:
  - For example,

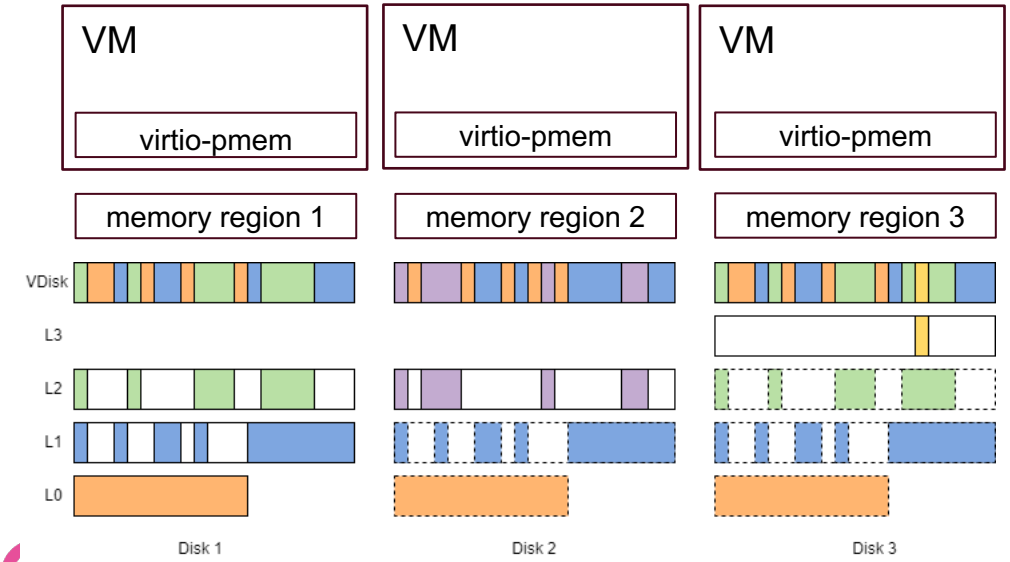
    mkfs.erofs –-tar=f –gzip layer0.erofs layer0.tgz
    mkfs.erofs –-tar=f –gzip layer1.erofs layer1.tgz
    …
    mkfs.erofs –-tar=f –gzip $layer_{n-1}$.erofs $layer_{n-1}$.tgz
    mkfs.erofs meta.erofs layer0.erofs layer1.erofs … $layer_{n-1}$.erofs



Case 2: layer-granularity FSDAX memory access

Problem

Snapshot-like FSDAX with traditional FSes can only do image-level de-duplication conveniently.

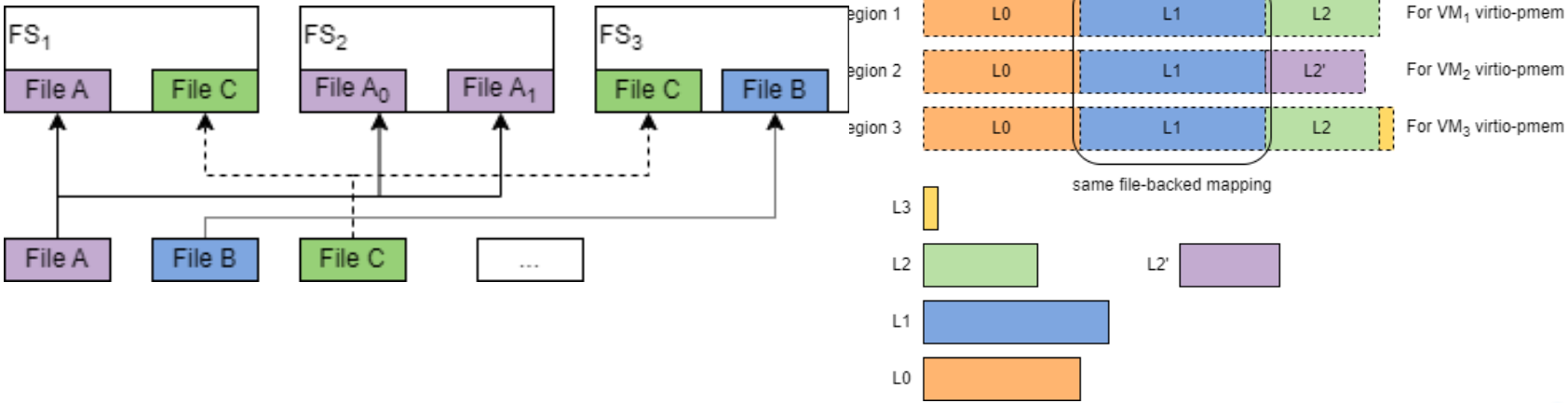


- Snapshot-like FSDAX memory access will have N-times extra memory overhead due to minor image update;
- It will become a burden to maintain another 4k-mapping table for snapshot-like FSDAX memory access in order to try to fix that.

Case 2: layer-granularity FSDAX memory access

Solution

- EROFS supports sub-image merging;
- Each PMEM memory region can be formed by multiple sub-images.



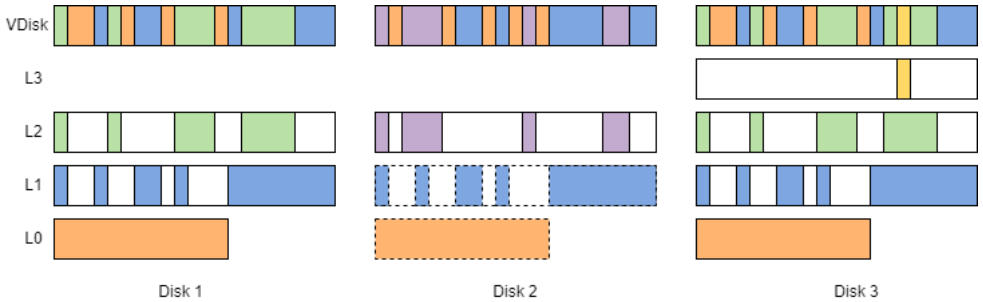
Case 3: runC page cache sharing

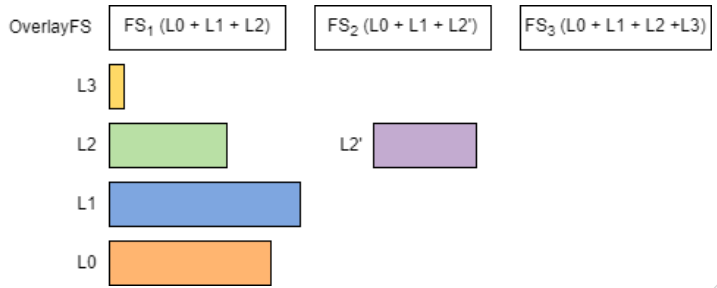
Problem

- Snapshot-like approaches with traditional FSes can hardly share page cache on RunC:

  Even compared to OverlayFS (layer-level page cache sharing);

- OverlayFS itself cannot share page cache on the same file across different layers:

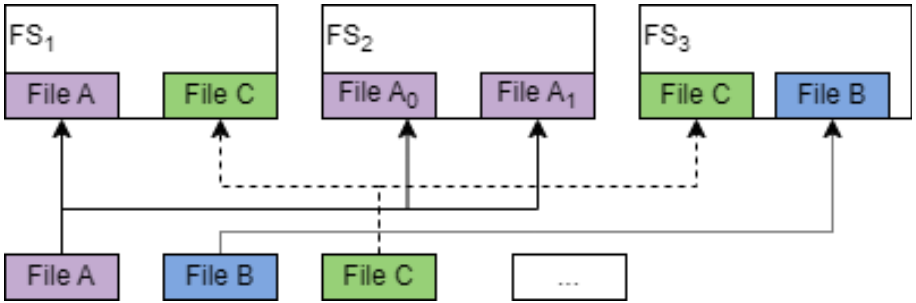




Case 3: runC page cache sharing

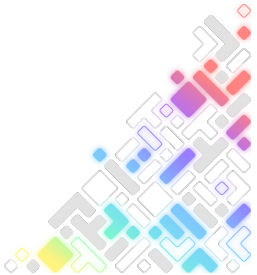
Solution

EROFS has in-house patches (upstreaming is still WIP) to support page caching sharing of files with the same data on a runC node;

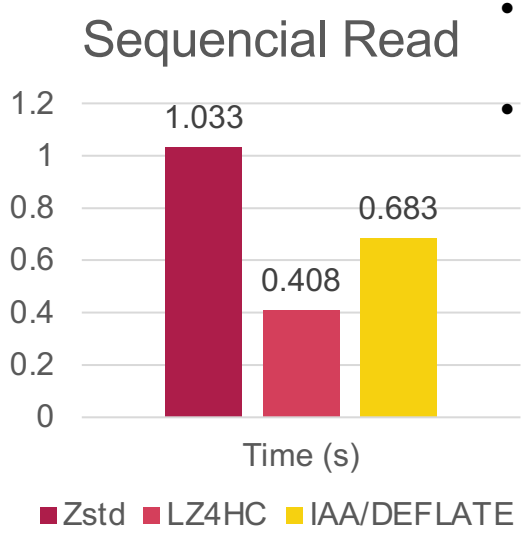


EROFS IAA preliminary testing

- The Intel® In-Memory Analytics Accelerator (Intel® IAA) is a hardware accelerator that provides very high throughput compression and decompression.
- It leverages the DEFLATE algorithm with a short sliding window (4 KiB).
- EROFS can use it to offload bulk data access.
- Currently I did some performance tests with Intel® QPL library, but
  - QPL doesn't have a dynamic library support https://github.com/intel/qpl/issues/31
  - QPL doesn't have pkgconfig support https://github.com/intel/qpl/issues/32
- In-kernel decompression support is still ongoing



EROFS IAA preliminary testing



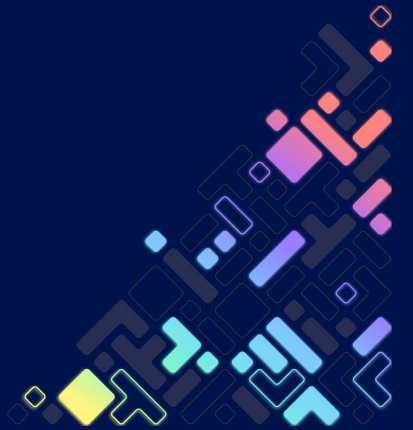
- IAA/DEFLATE accelerator performance is very close to LZ4 software decompression;
- IAA/DEFLATE images are **much smaller than LZ4HC**!

Algorithm	Image size
Zstd	298 MiB (312548986)
IAA (DEFLATE)	357 MiB (374890496)
LZ4HC	531 MiB (556879872)

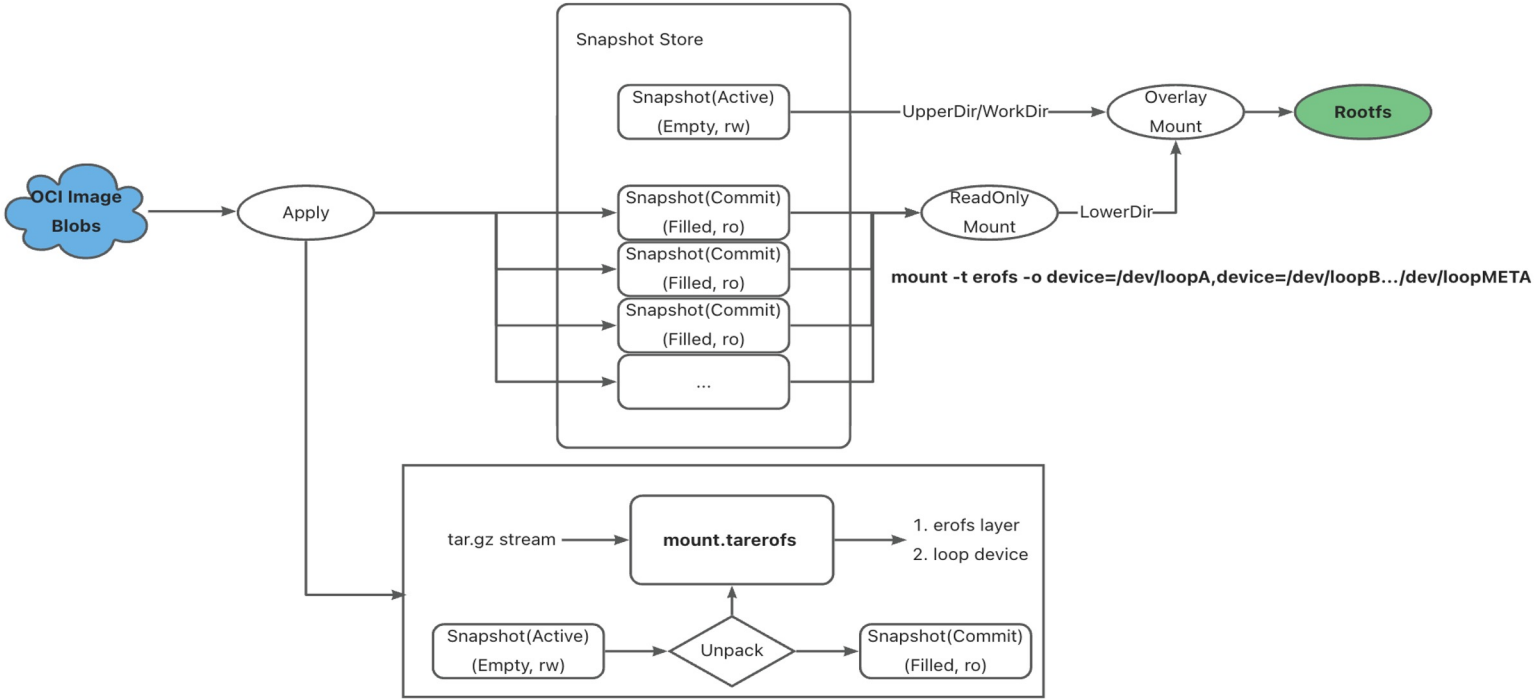
Test commands:

Zstd: time zstd –d –c –f enwik9.zstd > /dev/null
IAA: time fsck/fsck.erofs –extract enwik9.z.erofs
LZ4HC: time fsck/fsck.erofs –extract enwik9.lz4hc.erofs

Applications

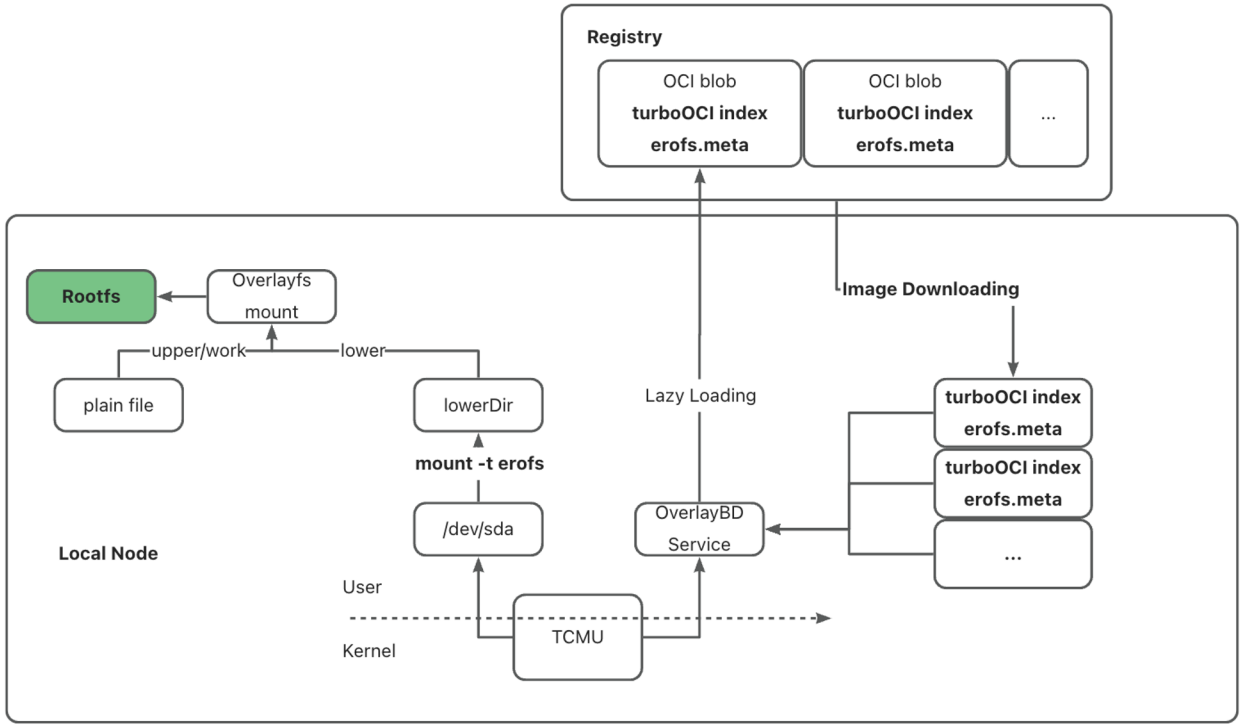


EROFS TarFS Snapshotter



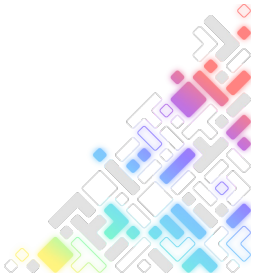
EROFS TarFS Snapshotter

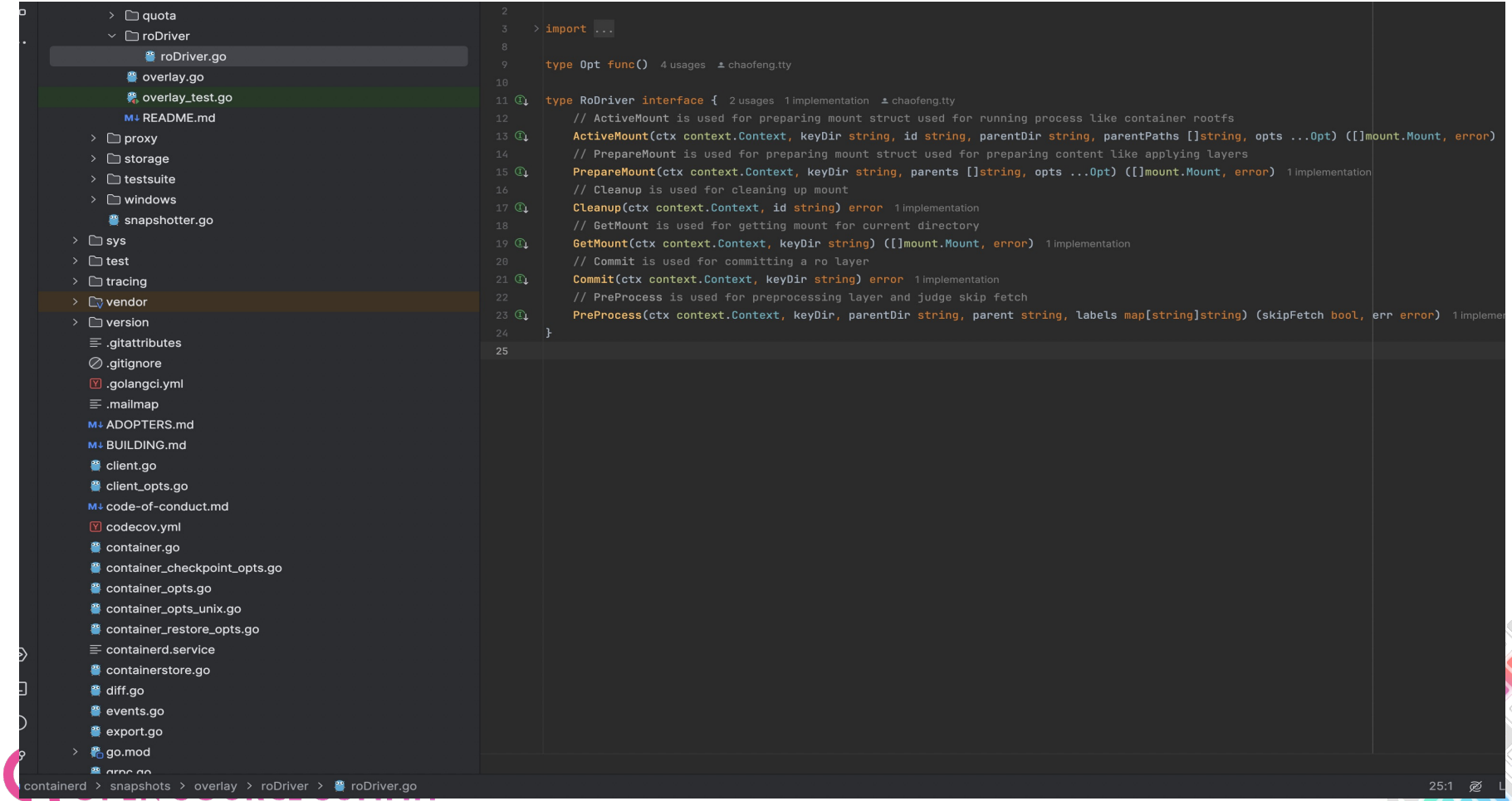
OverlayBD + EROFS



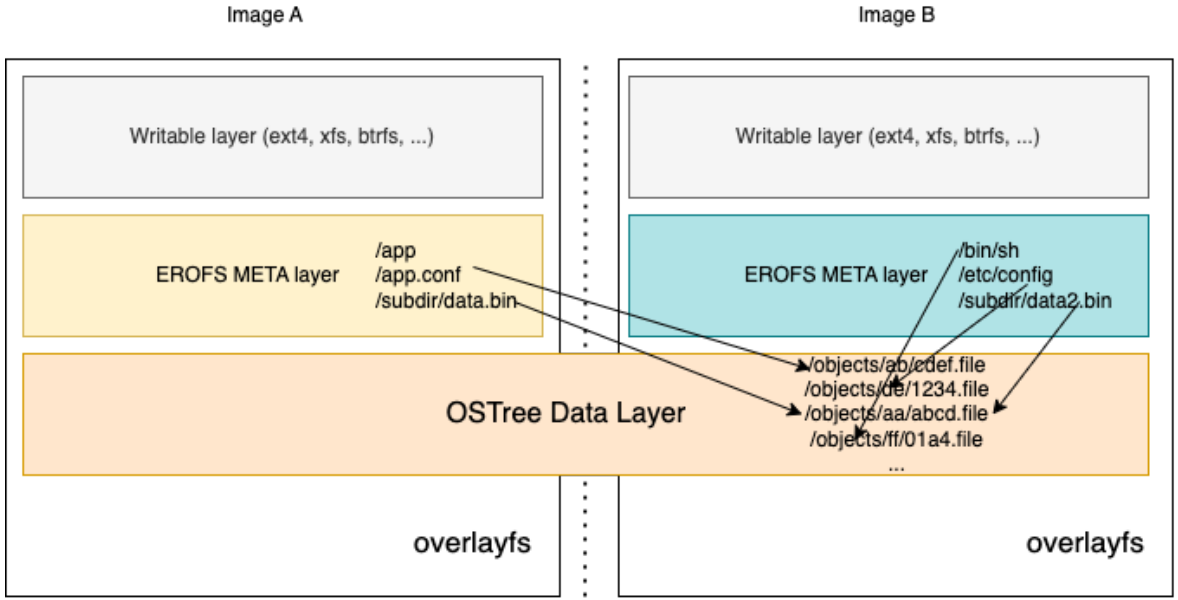
OverlayBD + EROFS

Demo time



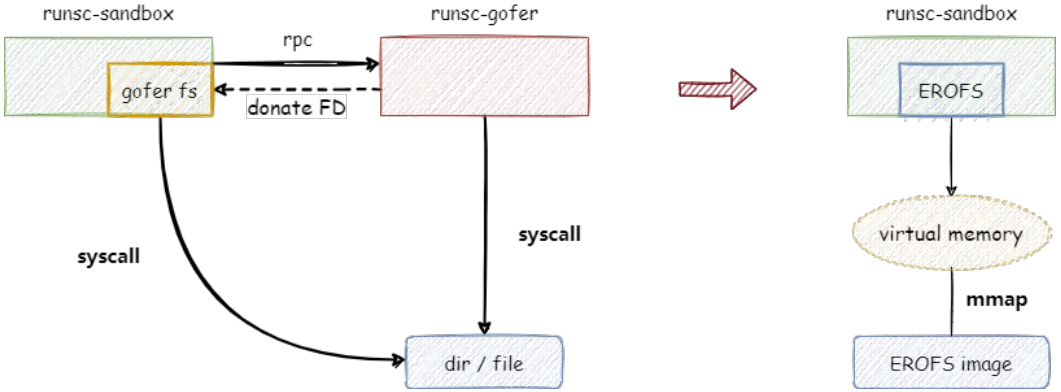


ComposeFS (w/ EROFS)



Composefs github: https://github.com/containers/composefs
Announcing composefs 1.0: https://blogs.gnome.org/alexl/2023/09/26/announcing-composefs-1-0/
FOSDEM 24 Composefs talk: https://fosdem.org/2024/schedule/event/fosdem-2024-3250-composefs-and-containers/

gVisor



Contributed by Tiwei Bie @ Ant Group
https://github.com/google/gvisor/pull/9486

Faster

Sandbox startup

Safer

Only raw images parsed by host

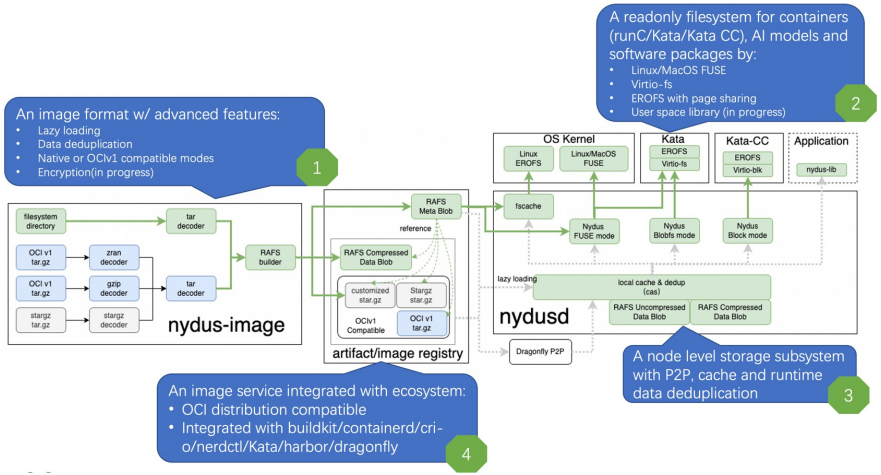
Easier

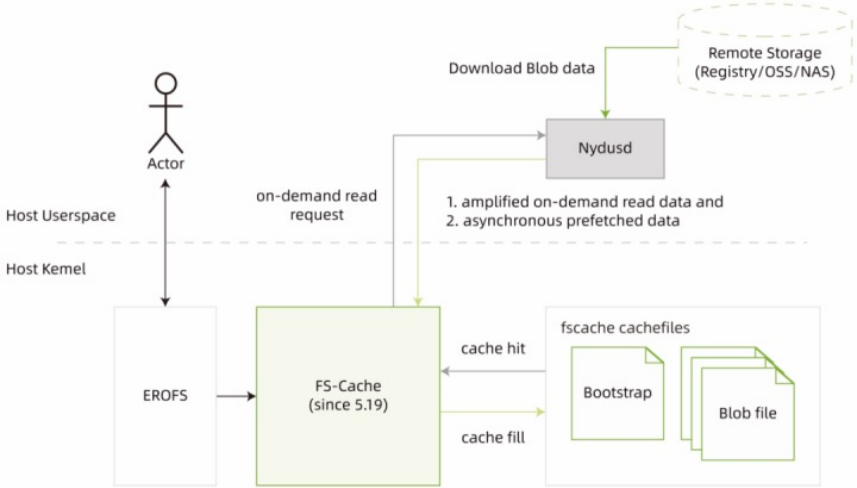
Read-only mapping is needed

50% CPU overhead decreased

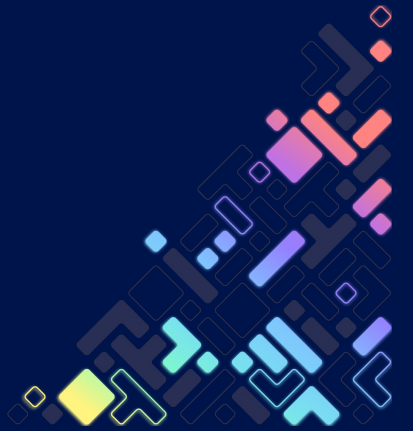
Nydus

- Dragonfly Nydus is a user-space example which uses in-kernel EROFS to leverage its functionality to do fast container image distribution like lazy pulling and data de-duplication across layers & images.
- Currently it can do lazy pulling for 1) Nydus/EROFS images, 2) (e)stargz images and 3) original OCI images with a minimal index (SOCI-like);



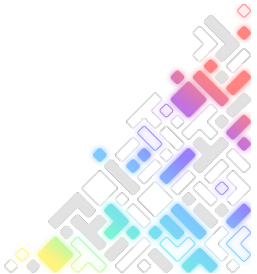


Roadmap



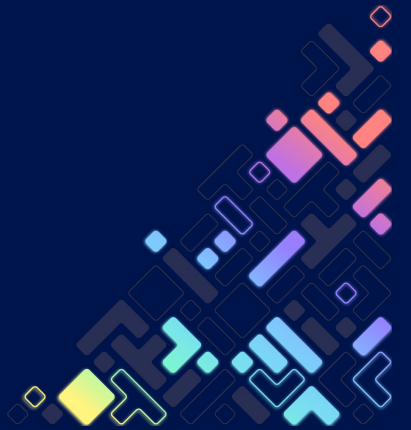
Roadmap

- Fully enable large folio support for compressed data;
- Preliminary Zstandard (de)compressor support;
- Upstream Intel QAT/IAA accelerator support;
- Upstream runC page cache sharing support;
- Preliminary EROFS Rust in-kernel adaption (EXPERIMENTAL, program for students):
  - https://summer-ospp.ac.cn/org/prodetail/241920019



Thank you!

https://erofs.docs.kernel.org



THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

NORTH AMERICA