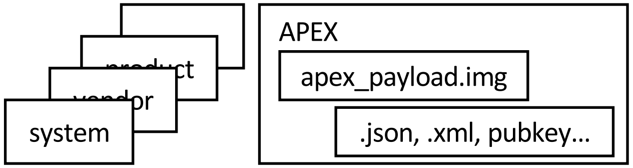
EROFS file system update and its future @ FOSDEM 23

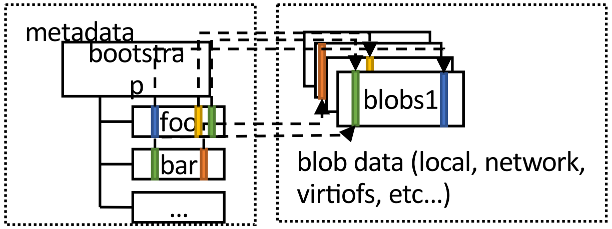
Xiang Gao <xiang@kernel.org>

What’s EROFS? Why EROFS?

- EROFS stands for Enhanced Read-Only File System (originally started in late 2017), available since Linux 4.19.
- It’s designed to be a generic high-performance read-only filesystem with a simple but effective core on-disk format design;
- It almost has the best performance among the current in-kernel read-only filesystems (as of v6.2);
- Kernel mountable as a seekable archival format replacement of traditional cpio and tar;
- Currently contributed by community lovers, Alibaba Cloud, ByteDance, Coolpad, Google, Huawei, OPPO, and more.
- Per-file LZ4 / LZMA (since 5.16) transparent data compression (as an option)
- Targeted for various high-performance read-only solutions:
  - System partitions & APEX for Android smartphone [1]
  - Other embedded systems (e.g. routers, IOT, ...)
  - LiveCDs (archiso, ...)
  - Container images (Nydus [2]) / app sandboxes
  - AI datasets
- Many useful features are actively under development [3]
  - Any suggestions or contributions are always welcome! ❤️



Android Smartphones



RAFS v6 (EROFS-compatible) container images

[1] https://source.android.com/docs/core/architecture/kernel/erofs
[2] https://github.com/dragonflyoss/image-service
[3] https://lore.kernel.org/linux-fsdevel/YqZNJpgQ+xLSHBqK@debian/

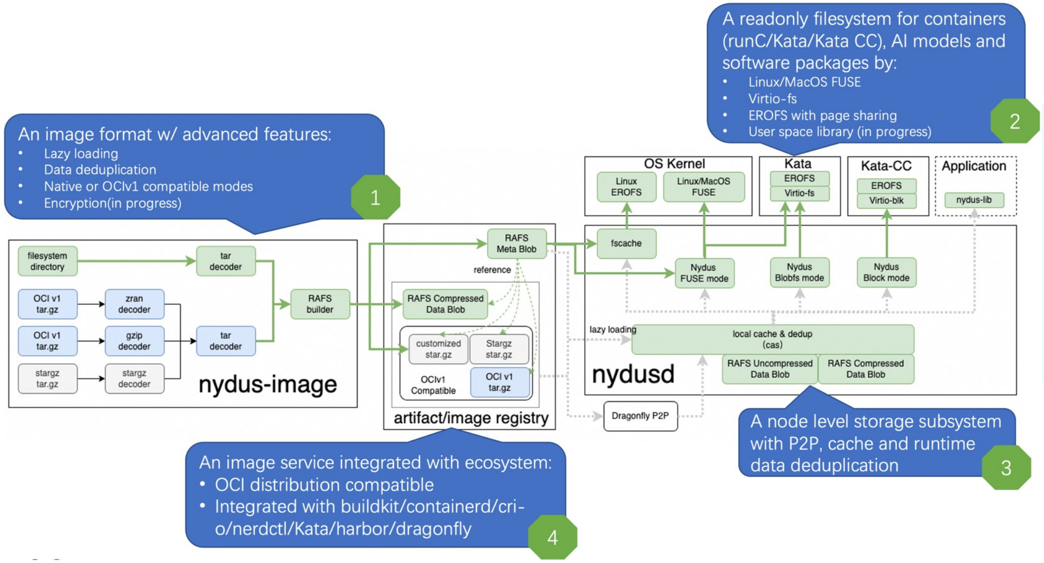
Use case: Android system partitions

- Android has several read-only partitions which behave as system fireware, which means “Android core can only be changed by way of an update”
- Benefits:
  - easy for vendors to ship/distribute/keep original signing (golden) images to each instance;
  - easy to roll back to the original shipped state or do incremental updates;
  - easy to check data corruption or do data recovery even in a very low level (e.g. hardware);
  - easy for real storage devices to do hardware write-protection;
  - and more;
- Why introducing EROFS [1]? Also APEXs and (even) APKs?

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

Use case: Container images —— 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 imags and 3) original OCI images with a minimal index (soci-like);
- For more details of Nydus itself, also see FOSDEM 23 **Nydus Image Service for Confidential Containers** @ Confidential Computing devroom


An image format w/ advanced features:
- Lazy loading
- Data deduplication
- Native or OCIv1 compatible modes
- Encryption(in progress)

1

A readonly filesystem for containers (runC/Kata/Kata CC), AI models and software packages by:
- Linux/MacOS FUSE
- Virtio-fs
- EROFS with page sharing
- User space library (in progress)

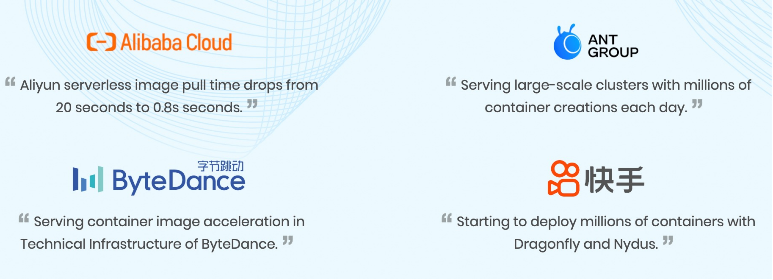
2

A node level storage subsystem with P2P, cache and runtime data deduplication

3

An image service integrated with ecosystem:
- OCI distribution compatible
- Integrated with buildkit/containerd/cri-o/nerdctl/Kata/harbor/dragonfly

4



“ Aliyun serverless image pull time drops from 20 seconds to 0.8s seconds. ”

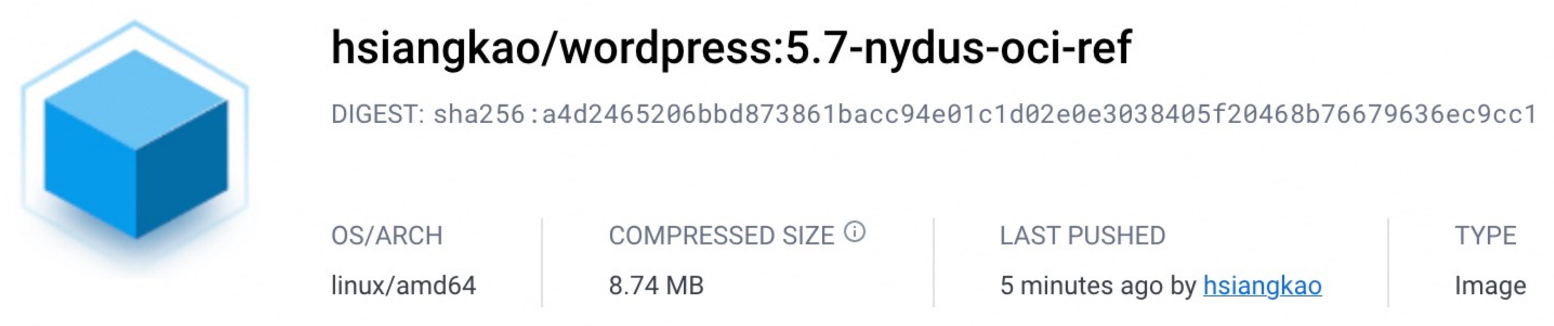
“ Serving large-scale clusters with millions of container creations each day. ”

“ Serving container image acceleration in Technical Infrastructure of ByteDance. ”

“ Starting to deploy millions of containers with Dragonfly and Nydus. ”

some partners which are landed Nydus + Dragonfly

Use case: Container images —— Nydus



hsiangkao/wordpress:5.7-nydus-oci-ref

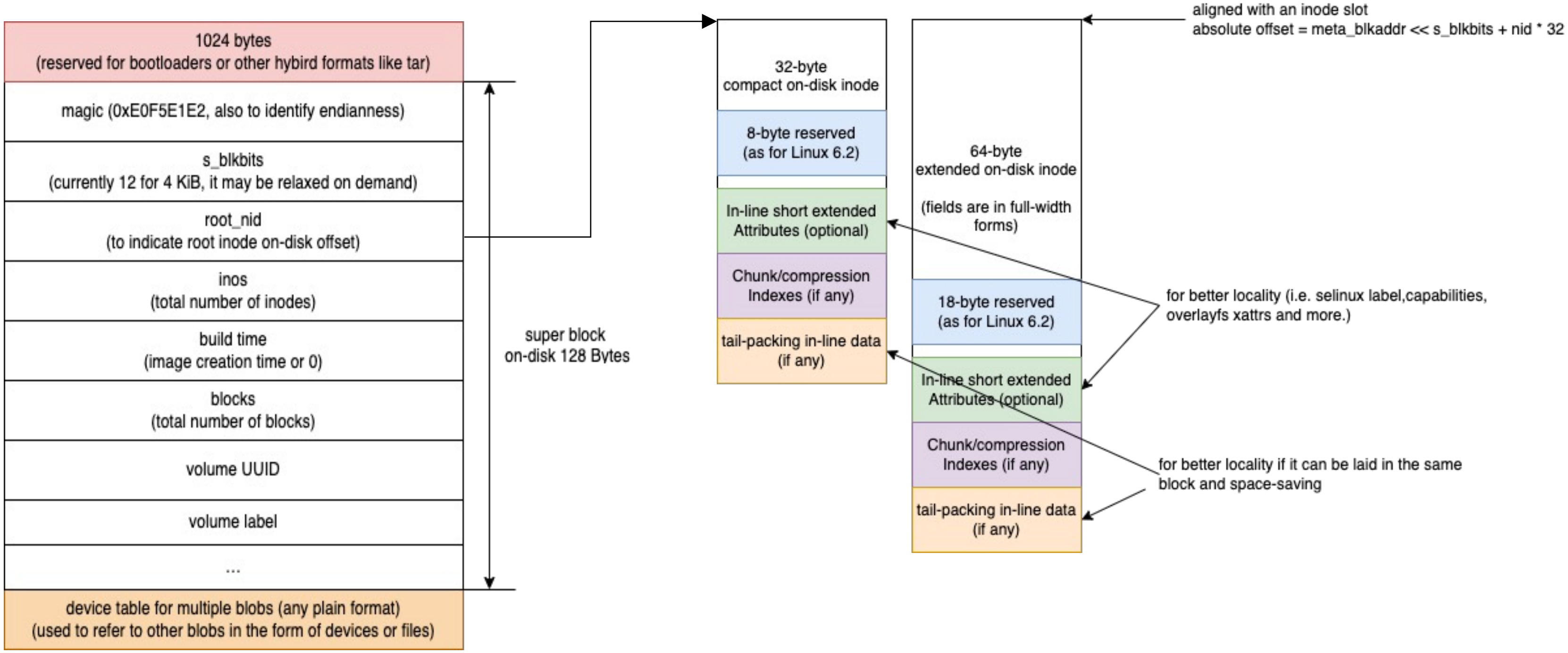
DIGEST: sha256:a4d2465206bbd873861bacc94e01c1d02e0e3038405f20468b76679636ec9cc1

OS/ARCH	COMPRESSED SIZE ⓘ	LAST PUSHED	TYPE
linux/amd64	8.74 MB	5 minutes ago by hsiangkao	Image

- EROFS running with original OCI + Nydus slim indexes

EROFS core internals in brief

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



1024 bytes
(reserved for bootloaders or other hybird formats like tar)

magic (0xE0F5E1E2, also to identify endianness)

s_blkbits
(currently 12 for 4 KiB, it may be relaxed on demand)

root_nid
(to indicate root inode on-disk offset)

inos
(total number of inodes)

build time
(image creation time or 0)

blocks
(total number of blocks)

volume UUID

volume label

...

device table for multiple blobs (any plain format)
(used to refer to other blobs in the form of devices or files)

super block
on-disk 128 Bytes

32-byte
compact on-disk inode

8-byte reserved
(as for Linux 6.2)

In-line short extended Attributes (optional)

Chunk/compression Indexes (if any)

tail-packing in-line data (if any)

64-byte
extended on-disk inode

(fields are in full-width forms)

18-byte reserved
(as for Linux 6.2)

In-line short extended Attributes (optional)

Chunk/compression Indexes (if any)

tail-packing in-line data (if any)

aligned with an inode slot
absolute offset = meta_blkaddr << s_blkbits + nid * 32

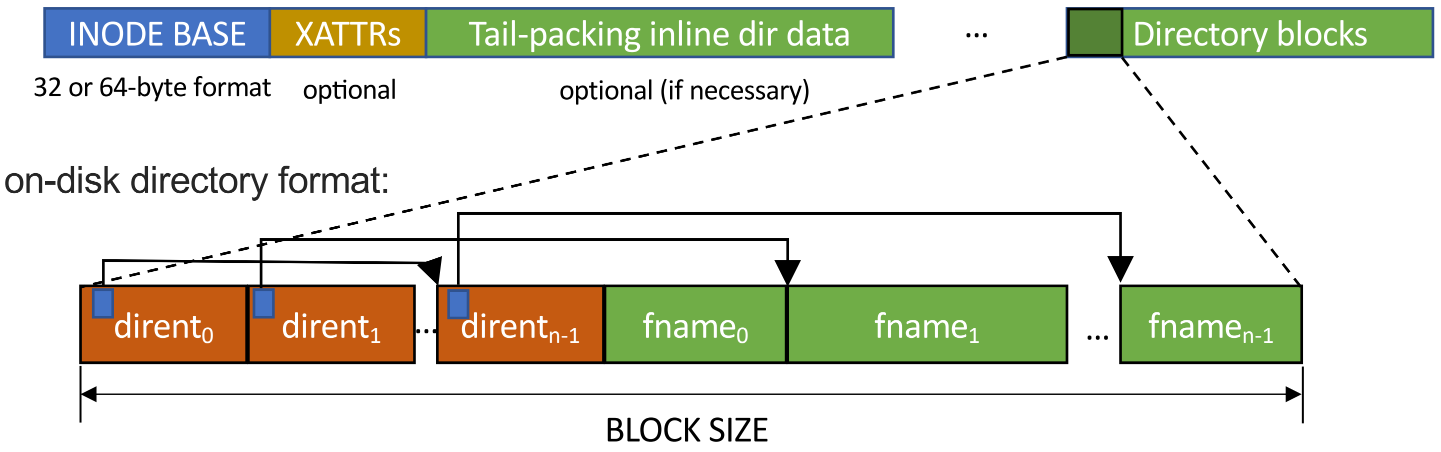
for better locality (i.e. selinux label,capabilities, overlayfs xattrs and more.)

for better locality if it can be laid in the same block and space-saving

EROFS core internals in brief

- **On-disk directory format**

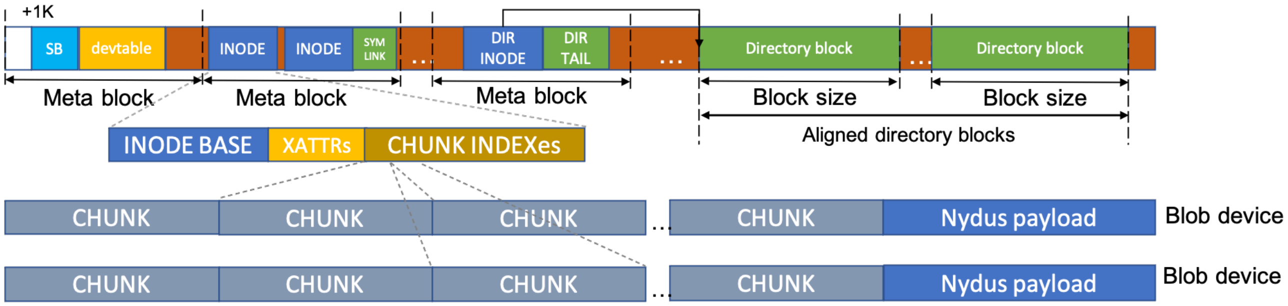
Directory files:



Filenames sorted in alphabetical order to improve performance by binary search.

EROFS core internals in brief

- Overview of Nydus use cases (since Linux v5.16)

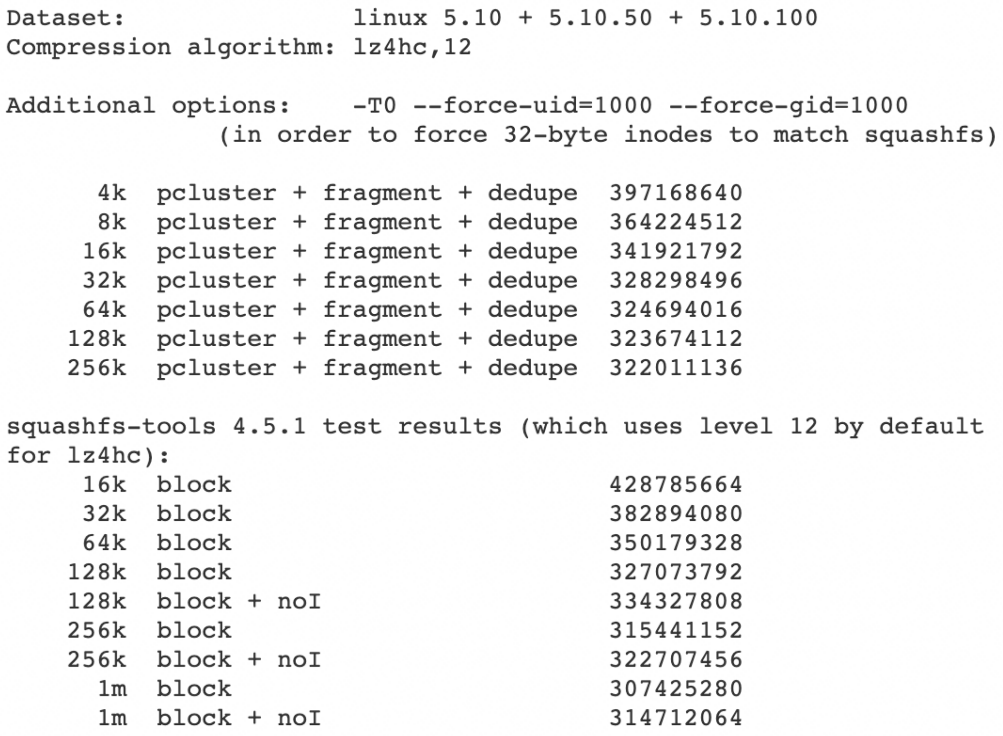


- Details of compressed data is somewhat not quite trivial, it could be referred from
    - EROFS Documentation <https://docs.kernel.org/filesystems/erofs.html>
    - EROFS ATC 19 Paper <https://www.usenix.org/conference/atc19/presentation/gao>

EROFS recent updates

- Chunk-based files —— sparse files and data-deduplicated plain files can be made.
- Multiple devices/blobs ——EROFS image can refer to other external data as well;
- EROFS over fscache (since v5.19, 2021-2022), which is already mentioned by some materials available online:
  - The evolution of the Nydus Image Acceleration
    - https://youtu.be/yr6CB1JN1xg
  - Introduction to Nydus Image Service on In-kernel EROFS @ OSSEU 2022
    - https://sched.co/15z3N
- Introduced a special inode (packed inode) for tail data (v6.1)
  - so that tail data or the whole of files can be deduped/compressed together
- Supported global compressed data deduplication by using rolling hash (v6.1)
- EROFS over fscache page cache sharing (WIP)

EROFS compressed data deduplication



```
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 inodes to match squashfs)

      4k  pcluster + fragment + dedupe  397168640
      8k  pcluster + fragment + dedupe  364224512
     16k  pcluster + fragment + dedupe  341921792
     32k  pcluster + fragment + dedupe  328298496
     64k  pcluster + fragment + dedupe  324694016
    128k  pcluster + fragment + dedupe  323674112
    256k  pcluster + fragment + dedupe  322011136

squashfs-tools 4.5.1 test results (which uses level 12 by default
for lz4hc):
     16k  block                         428785664
     32k  block                         382894080
     64k  block                         350179328
    128k  block                         327073792
    128k  block + noI                   334327808
    256k  block                         315441152
    256k  block + noI                   322707456
      1m  block                         307425280
      1m  block + noI                   314712064
```

https://git.kernel.org/pub/scm/linux/kernel/git/xiang/erofs-utils.git/commit/?id=990c7e38379547c4ffb98649913618eb76746844

EROFS future roadmap

- (self-contained) verification solution;
- (self-contained) data-deduplicated encryption solution;
- Fscache improvements together with Bytedance’s folks:
    - Failover;
    - Multiple daemons/dirs;
    - Daemonless.
- And more
    - https://lore.kernel.org/r/Y7vTpeNRaw3Nlm9B@debian

Thank you for listening!

- linux-erofs@lists.ozlabs.org
- https://nydus.dev
- IRC: hsiangkao @ oftc