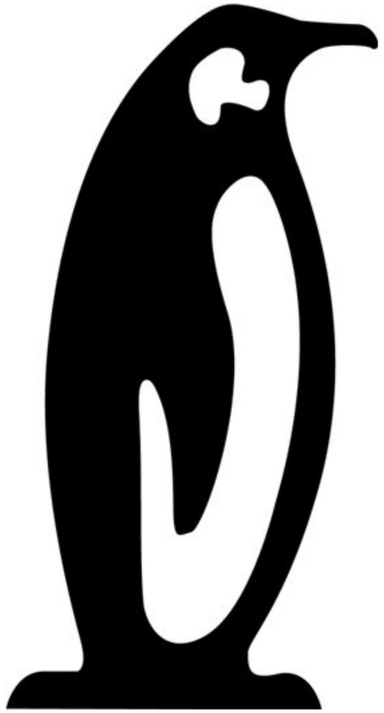


Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024

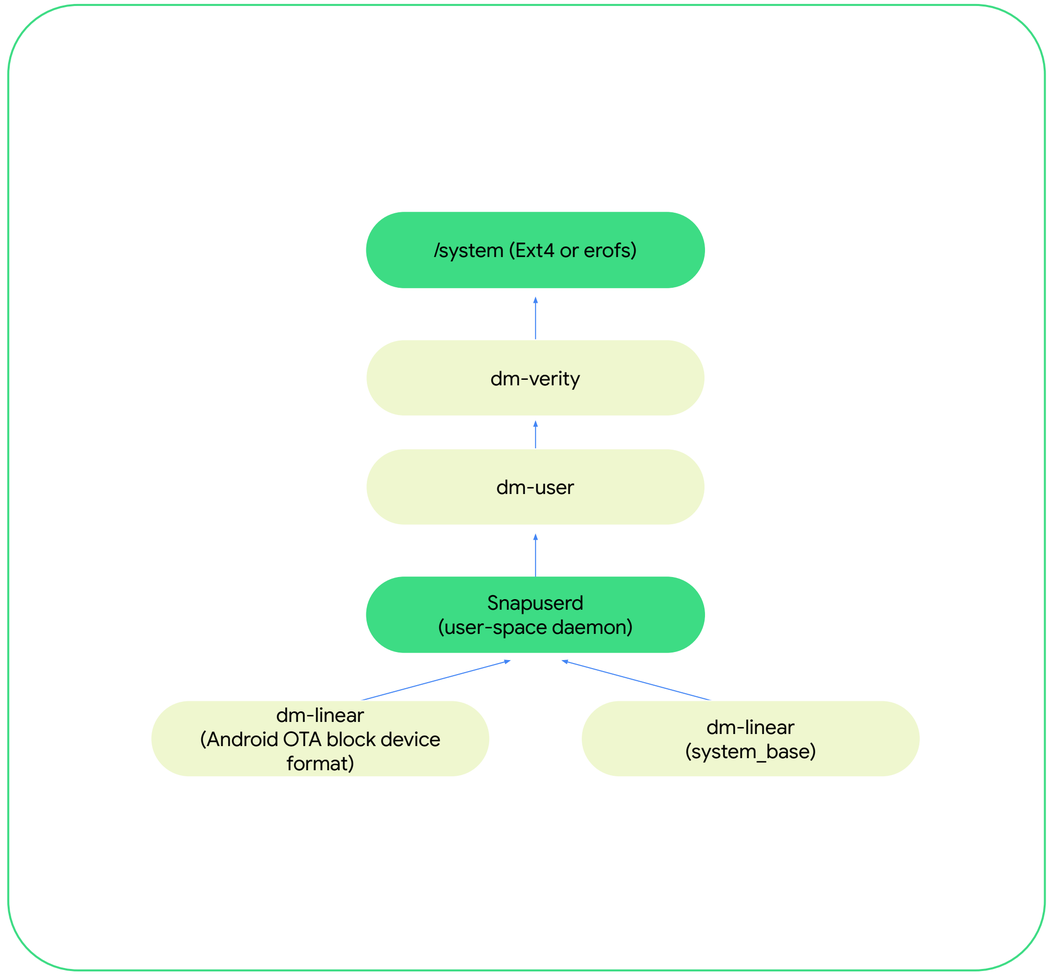
ublk based zero copy I/O - use case in Android

Akilesh Kailash (akailash@google.com)

LINUX PLUMBERS CONFERENCE | Vienna, Austria
Sept. 18-20, 2024

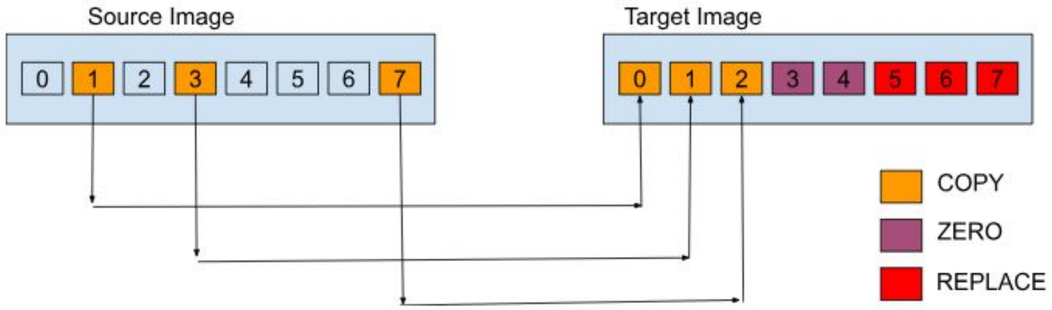
Android OTA - Storage Stack - Overview

- Userspace snapshots
  - dm-user: Out of tree kernel driver in ACK. Routes I/O request from verity to userspace daemon
  - Snapuserd daemon: Snapshot logic and snapshot merge
  - Root partition mounted off dm-user until snapshot merge is completed

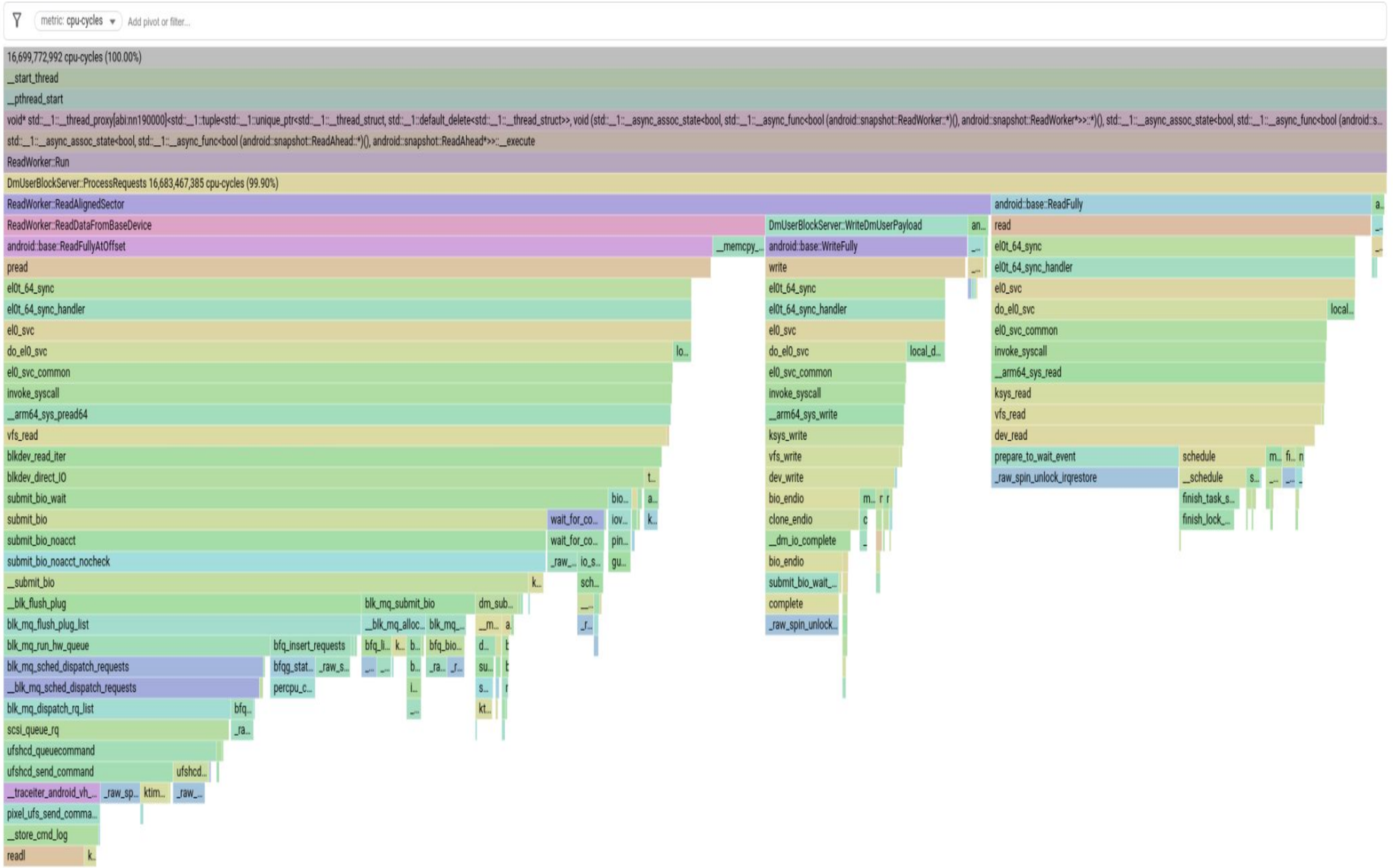


Android OTA Format

- Encodes three block-level operations:
  - ZERO: The destination block is zeroed.
  - COPY: The destination block is copied from a pre-existing block.
  - REPLACE: The destination block is replaced with new data. gz / lz4 compressed or could be uncompressed
  - XOR: XOR the destination block with source block.
- The metadata operations of the OTA format is stored in a block device.
  - Will be used by userspace daemon when snapshots are constructed

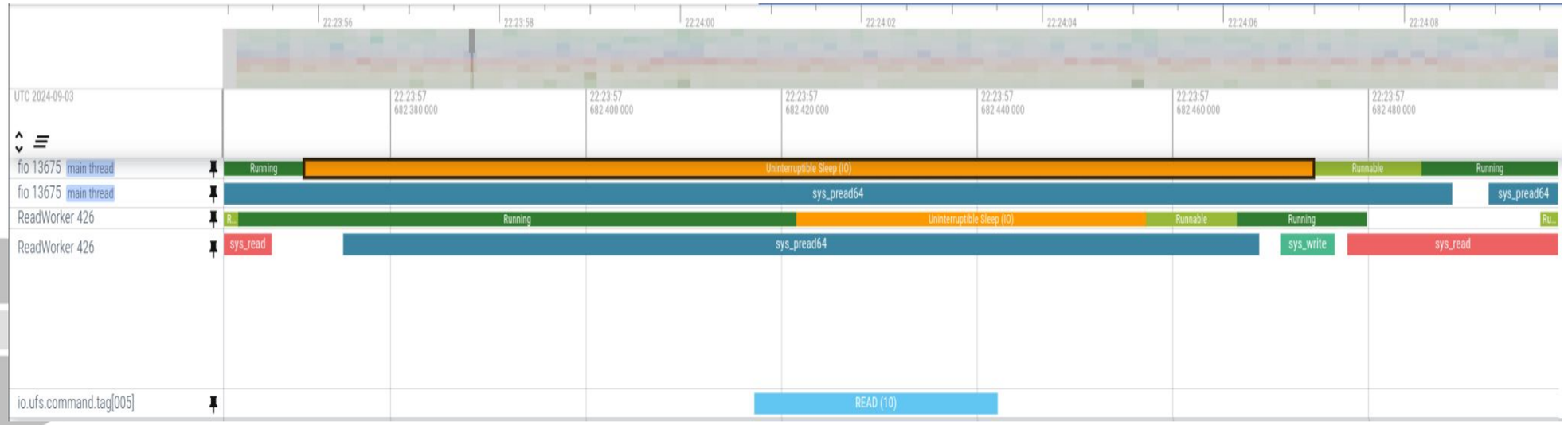


I/O Path for COPY operation - 4k block size



- COPY operation overhead in the I/O path
- Data moves back and forth between kernel and userspace
- ~50% of CPU cycles spent in reading data from source image into userspace.
- ~15% of CPU cycles spent in sending the data back to kernel from the userspace daemon
- Impacts OTA boot time and application I/O performance until snapshot merge is complete

Perfetto - I/O Path for COPY operation - a 4K I/O request

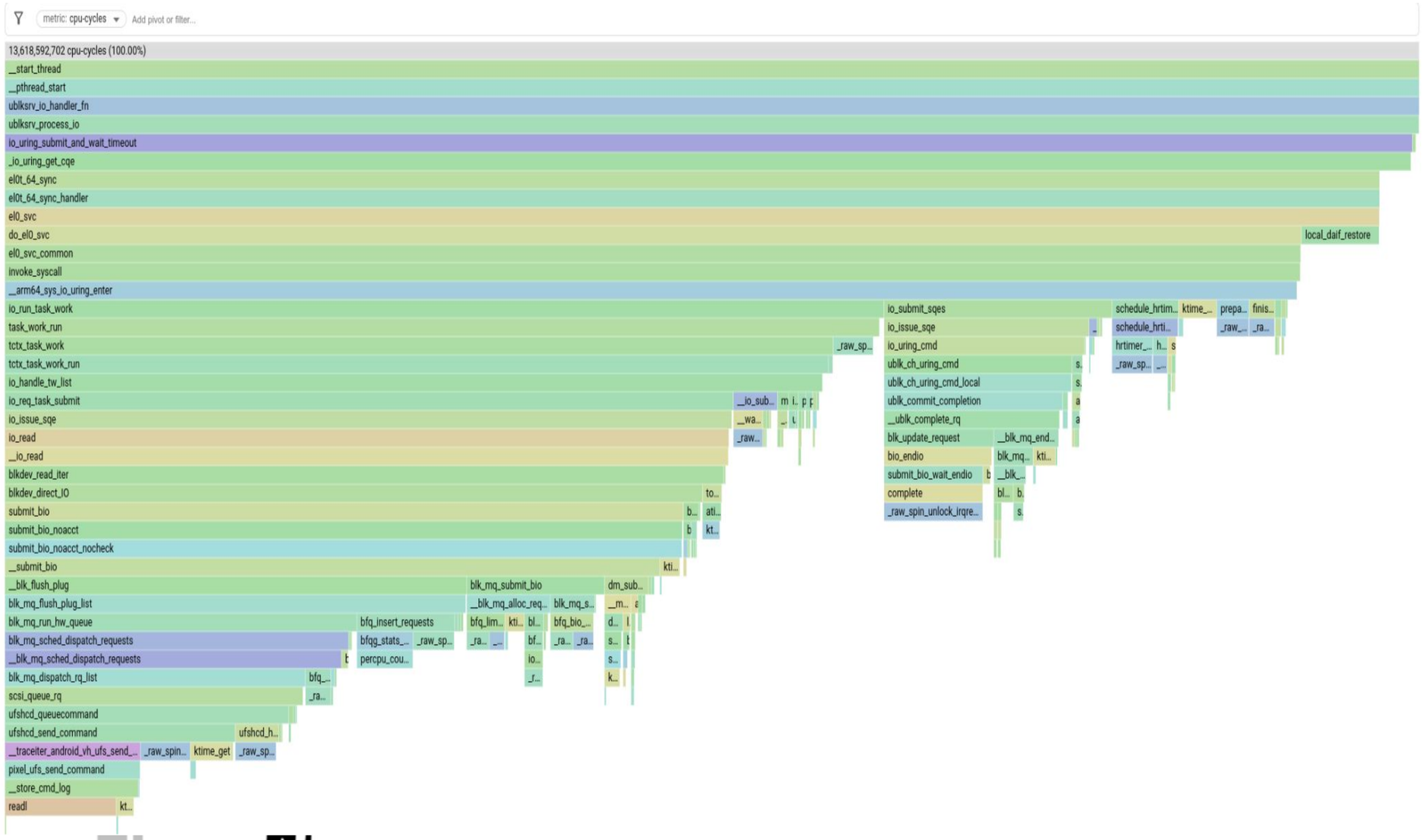


- ~100us time spent by application (fio) waiting for one 4k I/O to complete
- Userspace daemon (ReadWorker) is again blocked reading from underlying block device
- The actual I/O on underlying UFS is ~23us

ublk - userspace block device - zero copy

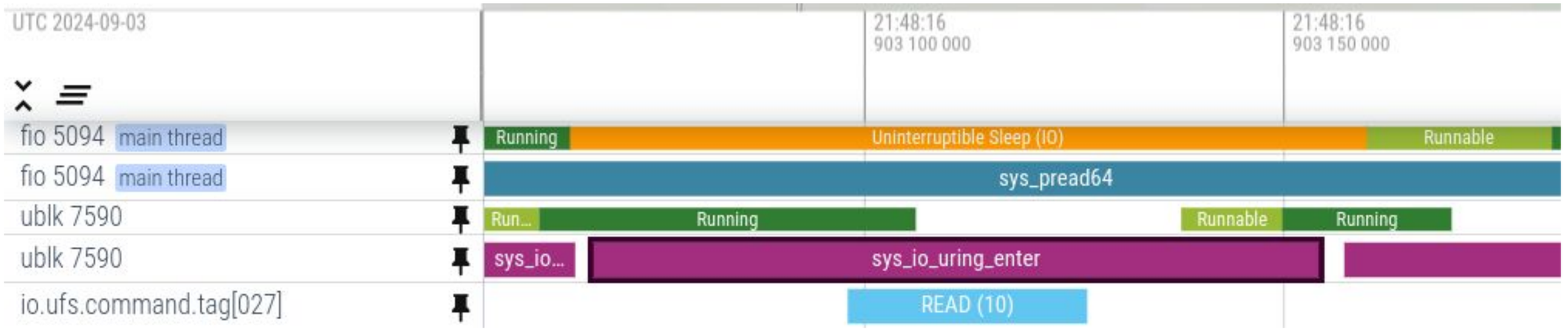
- Evaluate ongoing upstream patch - ublk zero copy based on io_uring providing sqe group buffer from Ming Lei
  - https://lore.kernel.org/io-uring/80d4150e-a4fe-4c05-be23-4ceebd40d7fd@gmail.com/T/#md440a9cebf9a1ed8e5cc204e6dcfdaa2f898e7a4
- ublk zero copy partially addresses some of the I/O path overhead - primarily for COPY operations in the OTA format
- Goal is to cut down the existing I/O path overhead for both COPY and REPLACE operations which accounts to 75% of operations
- V5 version of the patch evaluated on Pixel 6 running Android Mainline
  - Work in progress to move storage stack towards ublk replacing dm-user.
  - Zero copy effort helps in cutting down unnecessary data movement between userspace and kernel.
  - Primarily improving Android boot time and application performance until snapshot merge is complete.

I/O Path for COPY operation - 4k block size



- Data path is simple
- Two SQE entries - Lead SQE provides kbuf through io_import_group_kbuf
- One syscall completing entire I/O (io_uring_submit_and_wait)

Perfetto - I/O Path for COPY operation - a 4K I/O request



- No more additional I/O path overhead in the daemon
- Reduce CPU contention by more than 50%
- Android boot time

Caveats / Challenges

- How to handle REPLACE and ZERO operations through zero copy ?
- REPLACE operations are blocks which are compressed (lz4 or zstd) and stored in block device.
- The I/O path is similar to COPY operation but the data is retrieved from compressed block device.
- Currently, data is transferred to userspace, de-compressed in userspace and then data is transferred back to kernel.
- Similar I/O path overhead as observed on COPY operations.
- ublk zero copy would help but the data has to be decompressed in the kernel after importing kbuf from io_import_group_kbuf.
  - This will significantly help Full OTA which primarily consists of all REPLACE blocks.
  - Will further reduce the CPU contention post boot.
- Thoughts / Questions ?

Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024