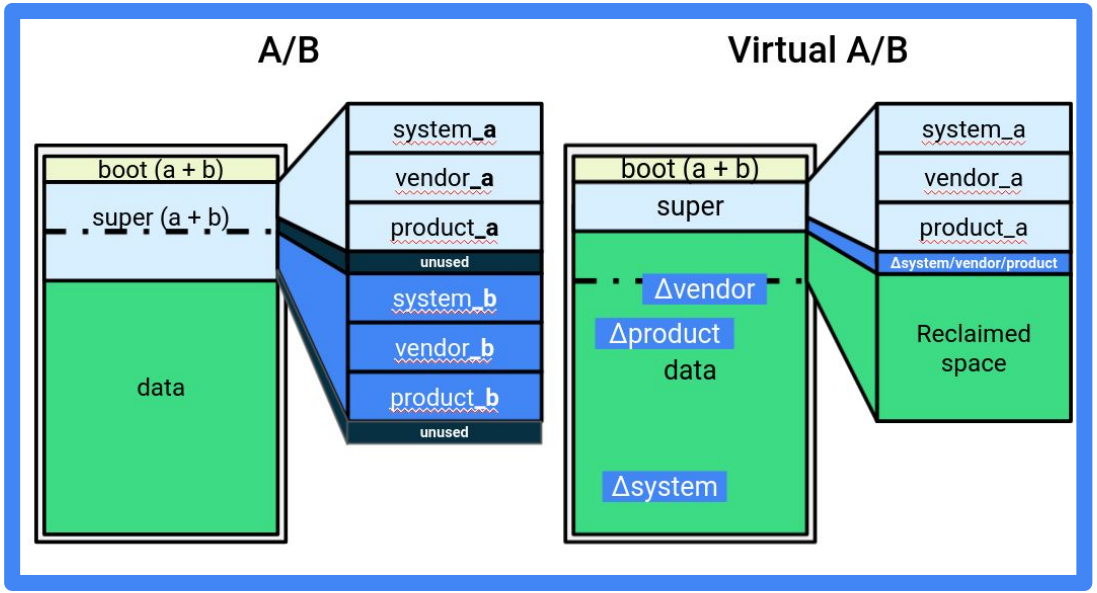
dm-snapshot in userspace

Akilesh Kailash <akailash@google.com>
David Anderson <dvander@google.com>

android

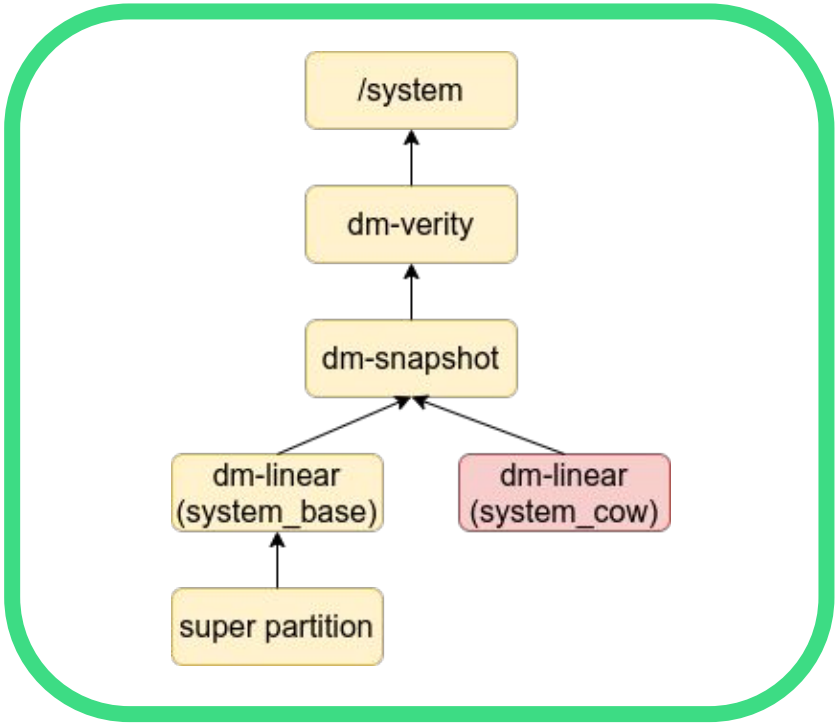
Behind the Scenes - Virtual A/B

- **Delta size** depends on the **update size** (copy-on-write) and can be computed in advance
- Space for **deltas** is **dynamically allocated** during an update (use free space in **super** and files in **/data**)



Usage of dm-snapshot (Android 11 (2020))

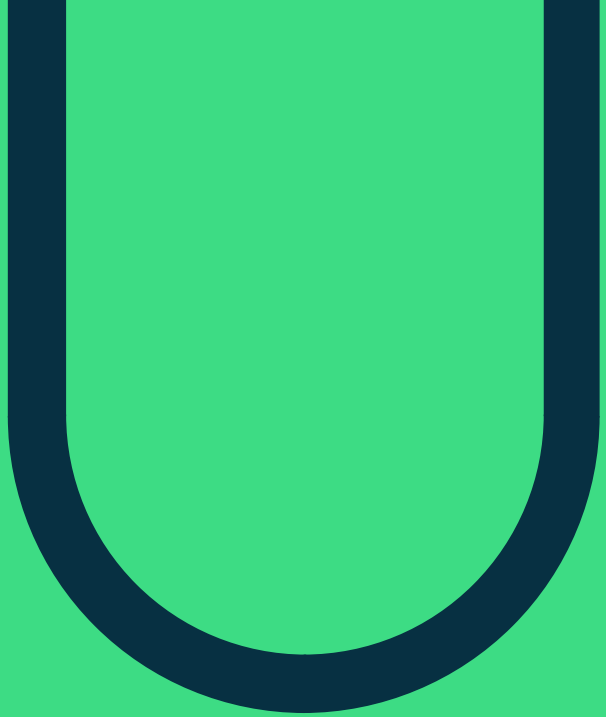
- dm-snapshot has a copy-on-write encoding native to the kernel
- No compression, no way to efficiently represent moves/copies or zeros
- If the OTA package encodes a "move" operation that totals 100MB of data, it will create 100MB of writes to system_cow.



Uncompressed Snapshots

- Feedback: Snapshots are too large!
- Reason: Snapshots effectively uncompress the OTA, causing them to use as much space as full A/B.
- Solution: Compress snapshots!

Compressed Snapshots



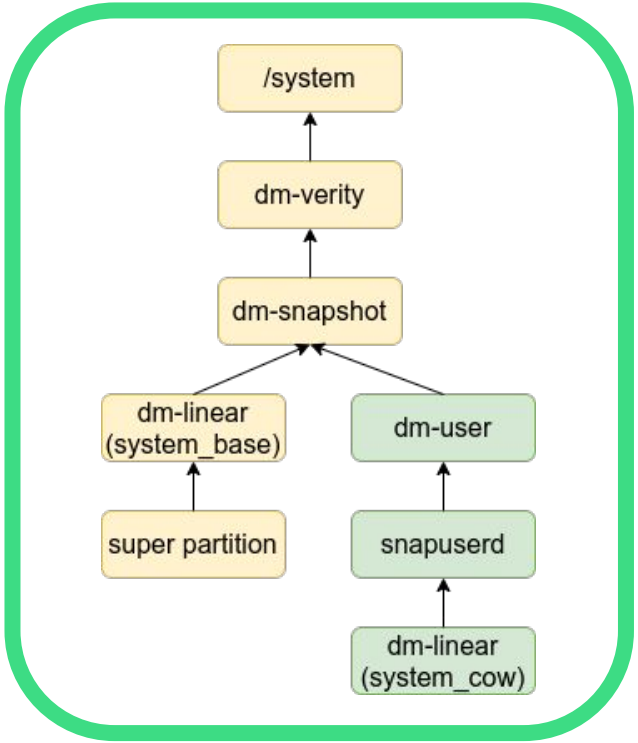
android

New Copy-on-Write Format (Android COW format)

- Encodes four 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-compressed into the COW.
  - XOR: The destination block is an XOR from a pre-existing block with the changed content stored in COW.

Usage of dm-snapshot (Android 12 (2021))

- dm-user - Kernel module, like FUSE but for block devices.
- snapuserd - Translates compressed snapshots back to the kernel.
- Kernel continues to make I/O requests as if it were using its native COW format.
- These requests are sent to userspace via dm-user, then translated into Android's COW format by snapuserd.
- dm-snapshot is read-only.



Space Reduction

- Compressed Snapshots save a huge amount of space compared to uncompressed Virtual A/B. Pixel samples:

	Full OTA (1.8G)	Incremental OTA (150MB)
VAB no compression	4173 MiB	3937 MiB
VAB compression	2328 MiB	492 MiB

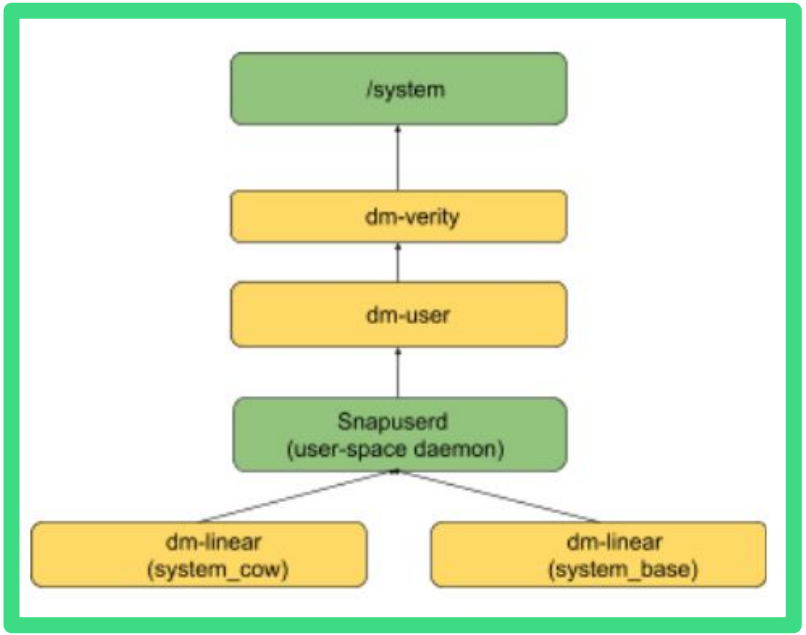
- Full OTAs are around 45% smaller.
- Incremental OTAs are around 8X smaller, depending on how many blocks were copied versus replaced.
- Smaller COWs make it easier to reduce the size of super and
give more space to users in /data.

Challenges

- Virtual AB Compression uses an Android-specific on-disk format to hold the diff between the A and B devices
  - COW format is too Android-specific for upstream
  - Fairly complex and ripe for future optimization
- Complexity of daemon is high as it intercepts the COW reads and writes occurring in the kernel, and implements them using the Android COW format. This translation makes daemon complex.
- Snapshot merge times are longer due to constant switching between user space and kernel.
- Boot time is impacted as every I/O to root partitions has to be served by the daemon.
- dm-snapshot is read-only. All writes to Android COW format are done in user space when OTA is downloaded.

Move Snapshot and Merge to user space

- Daemon will now be responsible for end-to-end merging.
- No more dependency with kernel dm-snapshots cutting down context switches between user-space and kernel during merge.
- root partition will be mounted off dm-user.
- Daemon will serve the I/O requests from dm-user



Performance and evaluation

- Prototype shows improvement in merge time performance

- Since partitions are mounted off dm-user, performance of dm-user is important as it can impact boot time
  - Performance of NBD (network block device) on boot time needs to be evaluated.
  - During boot process, I/O is carefully paused and resumed as snapuserd daemon is re-exec()'d to enable SEPolicy. We use device mapper "suspend" and "resume" functionality to achieve this transition.