RAID-Z Expansion

Matt Ahrens
Delphix & FreeBSD Foundation

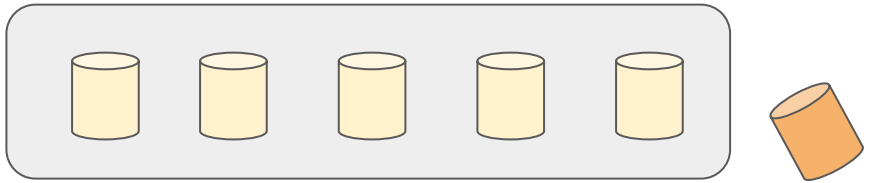
Don Brady
Klara Inc & iXsystems

Problem

```
+ zfs list test/fs
NAME      USED   AVAIL     REFER  MOUNTPOINT
test/fs  2.56G    331K     2.56G  /test/fs

+ zpool list -v test
NAME            SIZE  ALLOC    FREE  CKPOINT  EXPANDSZ    FRAG     CAP
test            4.50G  4.29G    215M        -         -     66%     95%
  raidz2        4.50G  4.29G    215M        -         -     66%   95.3%
    /var/tmp/1      -      -       -        -         -       -       -
    /var/tmp/2      -      -       -        -         -       -       -
    /var/tmp/3      -      -       -        -         -       -       -
    /var/tmp/4      -      -       -        -         -       -       -
    /var/tmp/5      -      -       -        -         -       -       -
```

Problem



You have a full RAID-Z pool.

Solution

Add another disk, expanding pool’s capacity.

Solution

```
+ zpool attach test raidz2-0 /var/tmp/6
+ zpool status test
  pool: test
 state: ONLINE
raidz expand: Expansion of vdev 0 in progress since Wed Jun  9 16:36:19 2021
    444M copied out of 4.29G at 22.2M/s, 10.12% done, 0h2m to go
config:
      NAME             STATE     READ WRITE CKSUM
      test             ONLINE       0     0     0
        raidz2-0       ONLINE       0     0     0
          ...
          /var/tmp/5   ONLINE       0     0     0
          /var/tmp/6   ONLINE       0     0     0

+ zfs list test/fs
NAME      USED   AVAIL     REFER  MOUNTPOINT
test/fs  2.56G    323K     2.56G  /test/fs
+ zpool list -v test
NAME             SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP
test            4.50G  4.29G   215M        -         -    60%    95%
  raidz2        4.50G  4.29G   215M        -         -    60%  95.3%
```

Solution

```
+ zpool wait test -t raidz_expand
+ zpool status test
  pool: test
 state: ONLINE
raidz expand: Expansion of vdev 0 copied 4.27G in 0h3m, completed on Wed Jun  9 16:39:31 2021
config:

        NAME             STATE     READ WRITE CKSUM
        test             ONLINE       0     0     0
          raidz2-0       ONLINE       0     0     0
            ...
            /var/tmp/5   ONLINE       0     0     0
            /var/tmp/6   ONLINE       0     0     0

+ zfs list test/fs
NAME      USED  AVAIL     REFER  MOUNTPOINT
test/fs  2.56G   610M     2.56G  /test/fs
+ zpool list -v test
NAME             SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP
test            5.50G  4.29G  1.21G        -         -    51%    77%
  raidz2        5.50G  4.29G  1.21G        -         -    51%  78.0%
```

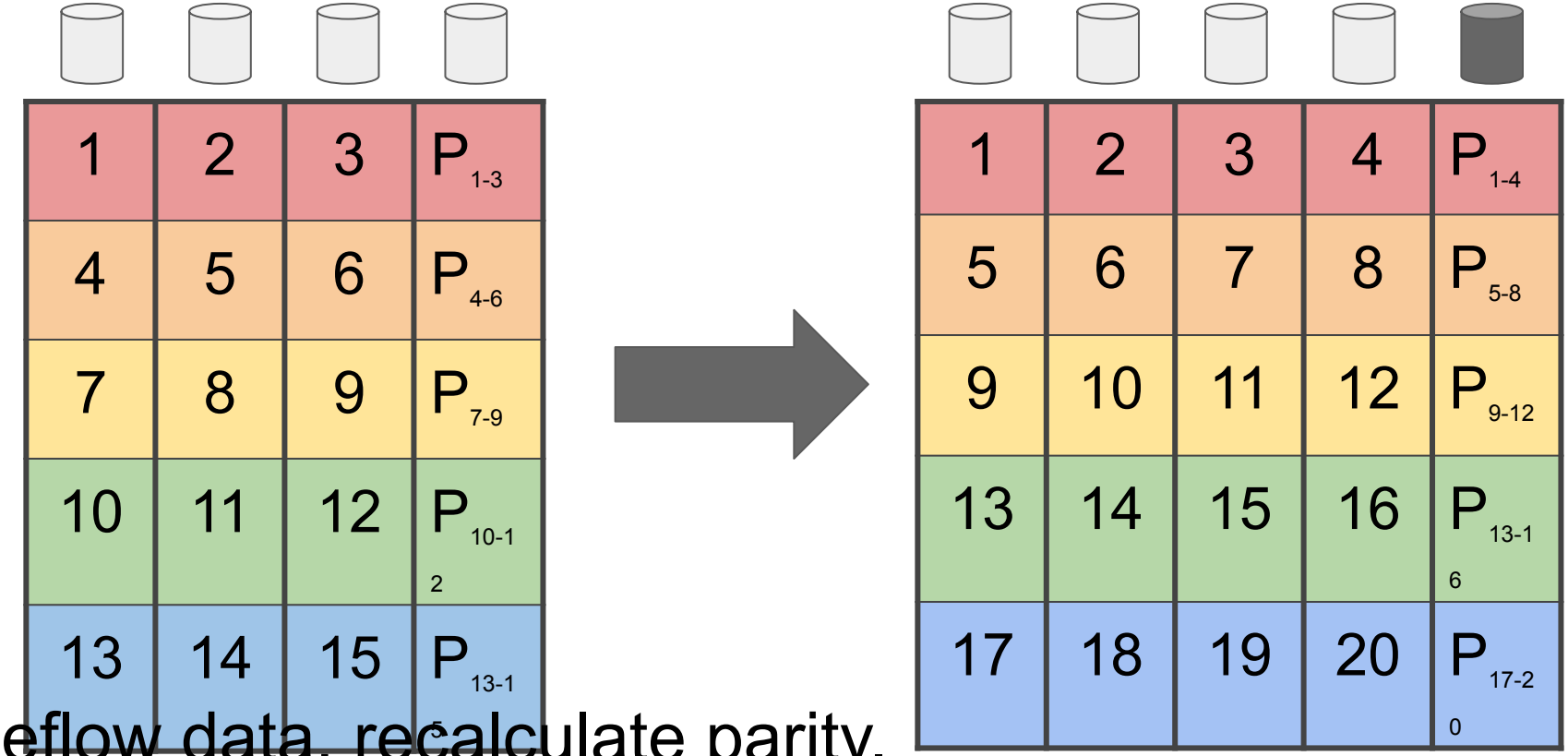
Reminder:
USED/AVAIL
is not
ALLOC/FREE!

How does it work?

What does the on-disk state look like after expansion completes?

p.s. why did it take 6 years?

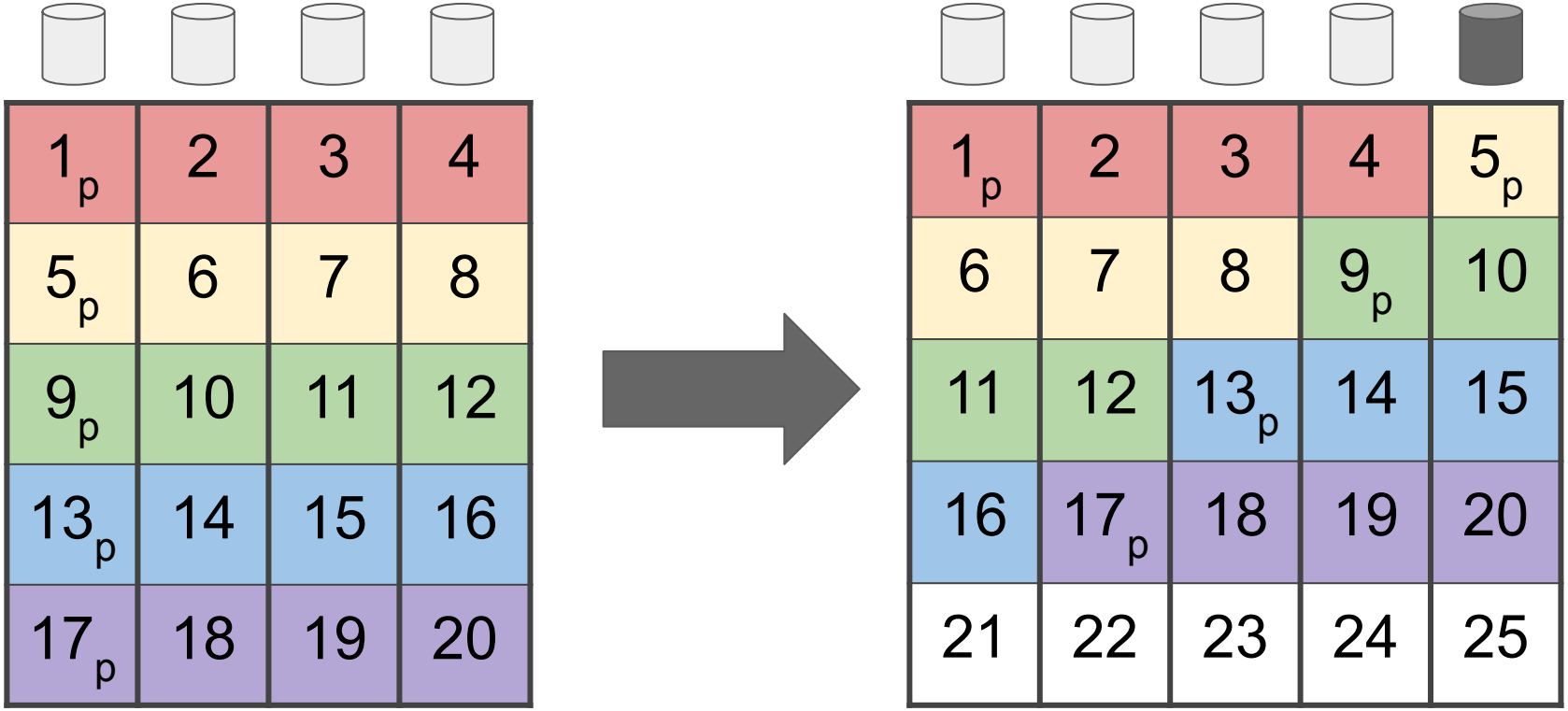
How does traditional RAID 4/5/6 do it?



Reflow data, recalculate parity.

(Color indicates parity group / stripe.)

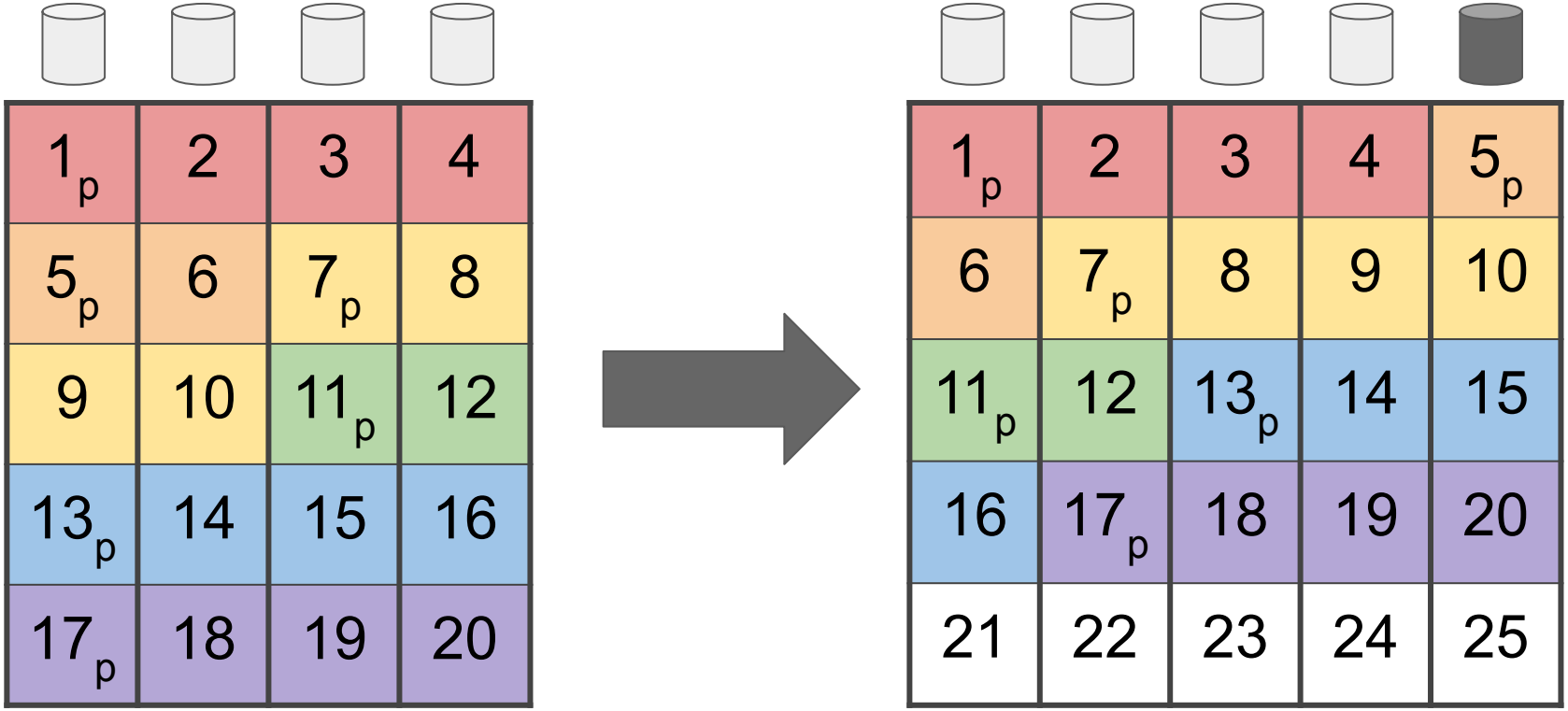
RAID-Z Expansion: Reflow



Reflow data and parity blocks.

(Color indicates parity group / stripe.)

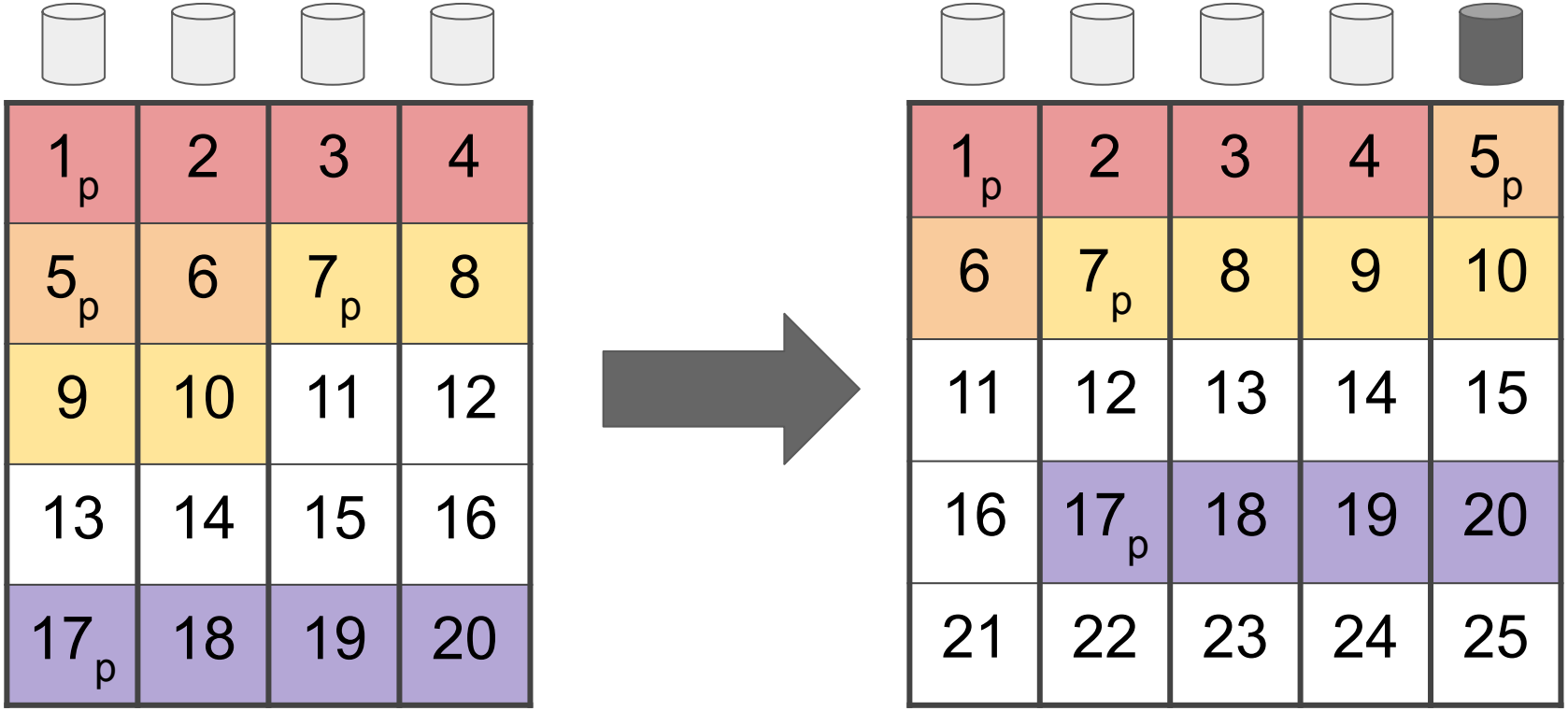
RAID-Z Expansion: Reflow doesn't care where parity is!



Reflow data and parity blocks.

(Color indicates parity group / stripe.)

RAID-Z Expansion: Reflow copies allocated data



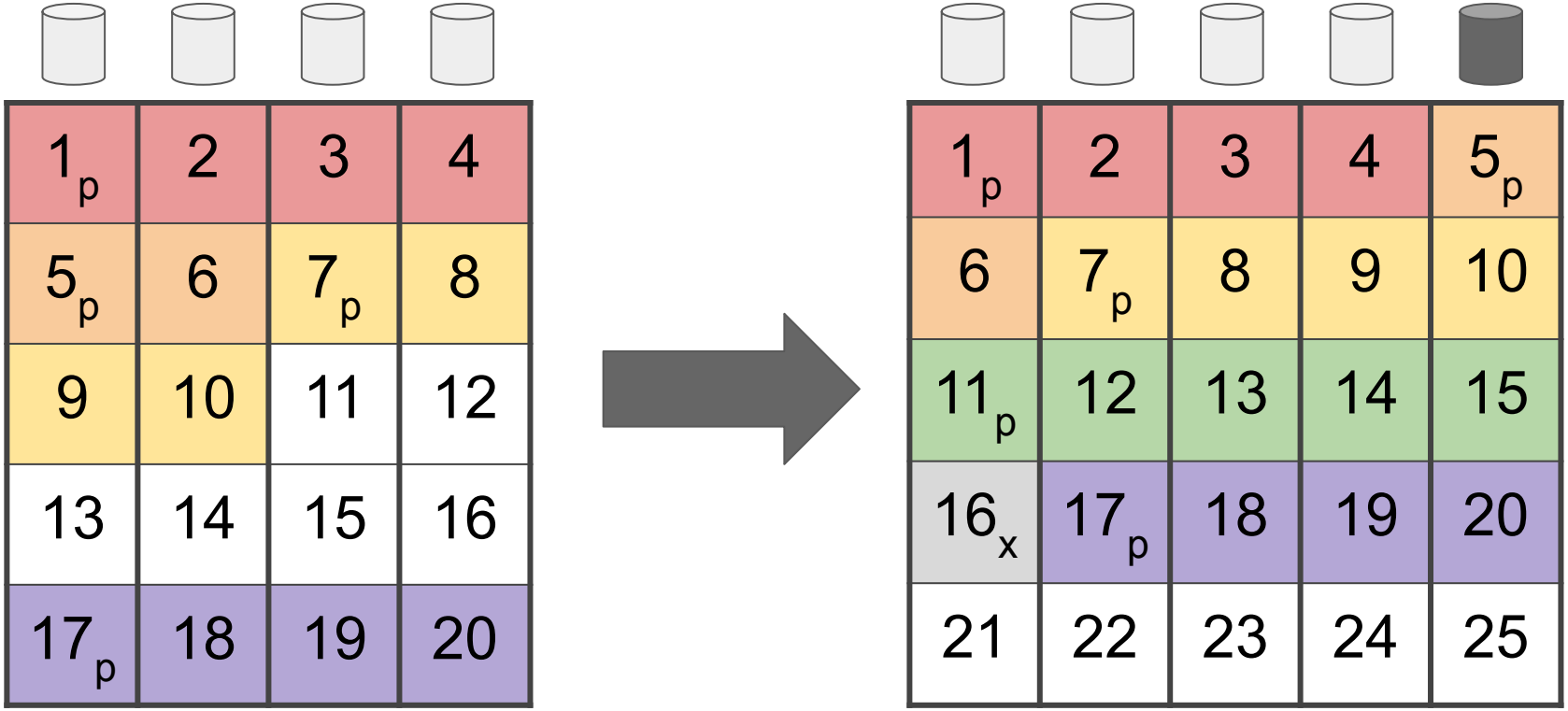
Reflow data and parity blocks.

(Color indicates parity group / stripe.)

Reflow works!

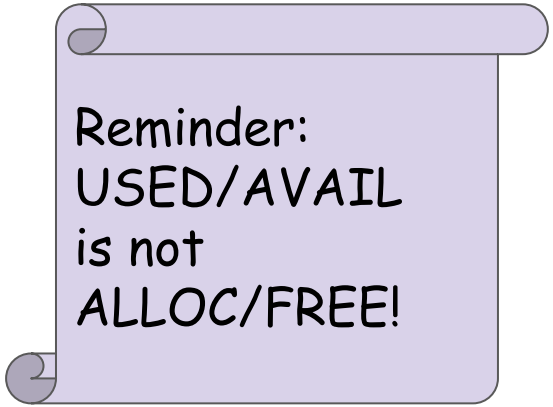
- Doesn’t change (or even read) block pointers
- Reads / Writes sequentially
- Spacemaps tell us what we need to copy
- Each logical stripe is independent
  - Don’t need to know where parity is
  - Segments still on different disks, so redundancy is preserved
  - (contraction couldn’t work this way)

RAID-Z Expansion: new writes, new stripe width



Color indicates parity group (logical stripe)

Logical vs Physical stripe width



- After conversion, physical stripe width is 5
- Old blocks still have logical stripe width of 4
- New blocks have logical stripe width of 5
  - Improved data : parity ratio (4:1 instead of 3:1)
- When reading, need to know logical stripe width
  - Use block's birth time (+ expansion time) to determine
- Assumed data:parity ratio doesn't change, so space accounting may show slightly lower than expected usage (per "zfs list" (USED), "du", "ls -s", etc)

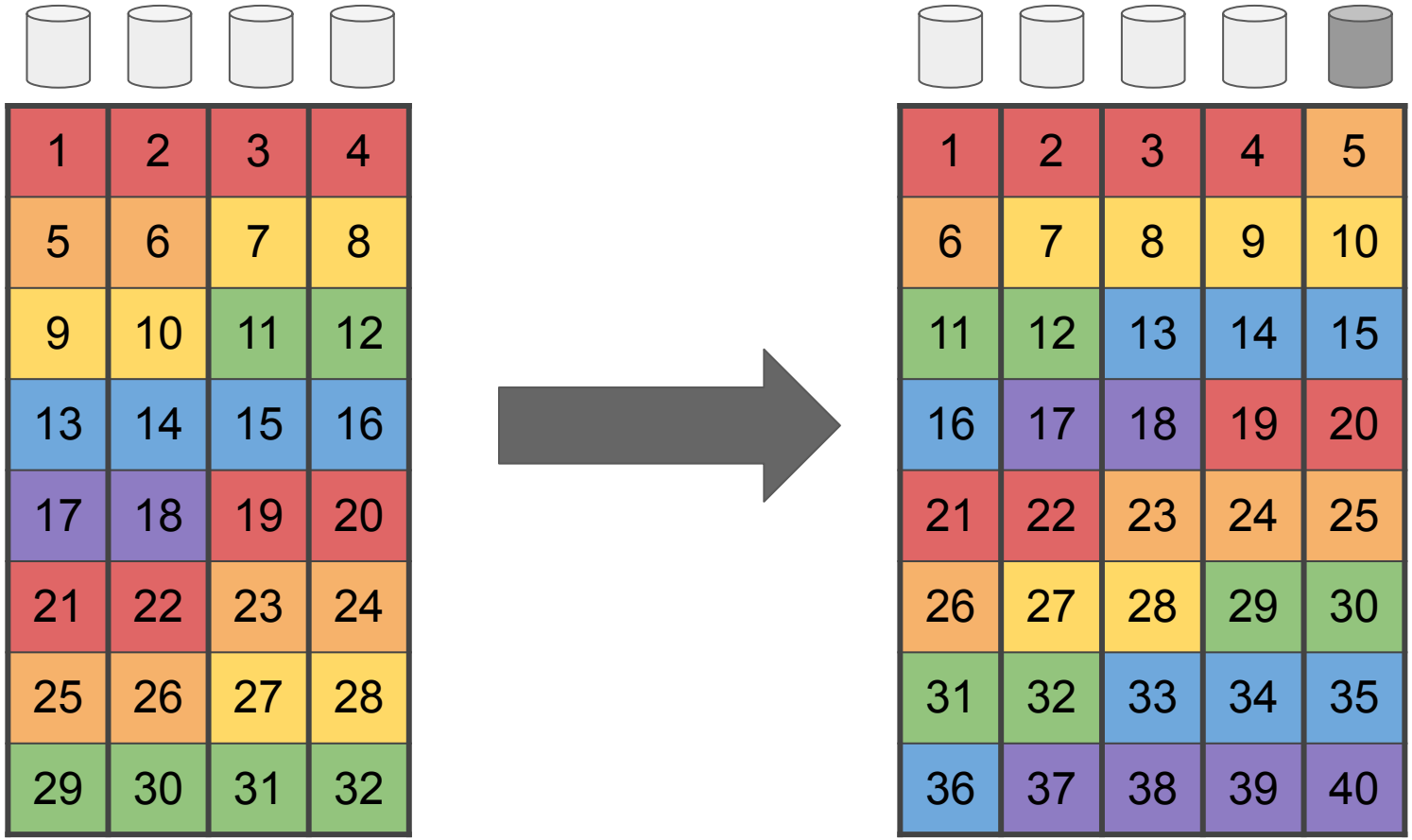
How does it work?

How do we get from initial state to end state?

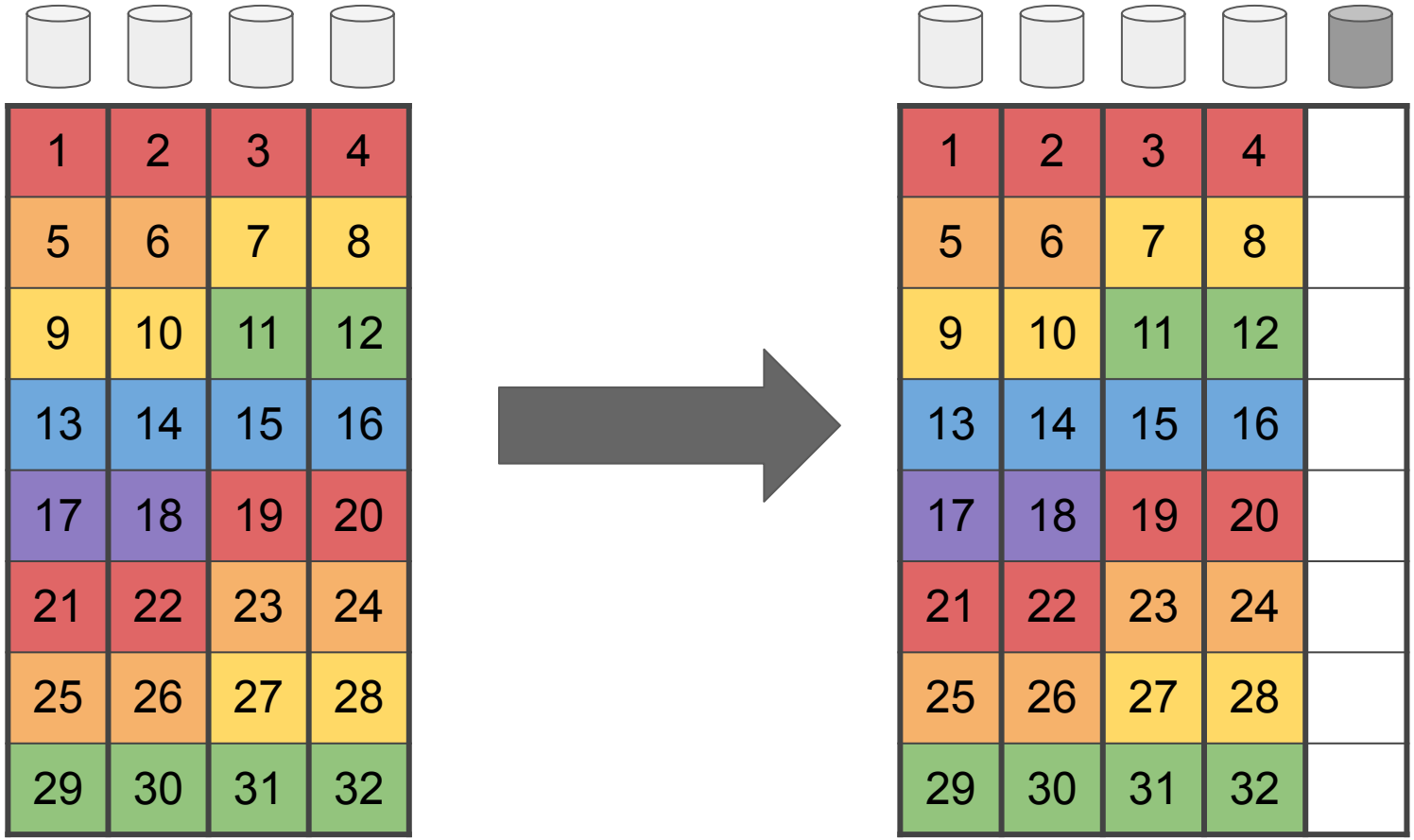
Incrementally, since we can’t read the entire pool into RAM (or unused disk space) and then write it back out again.

Online, while concurrently processing normal reads/writes.

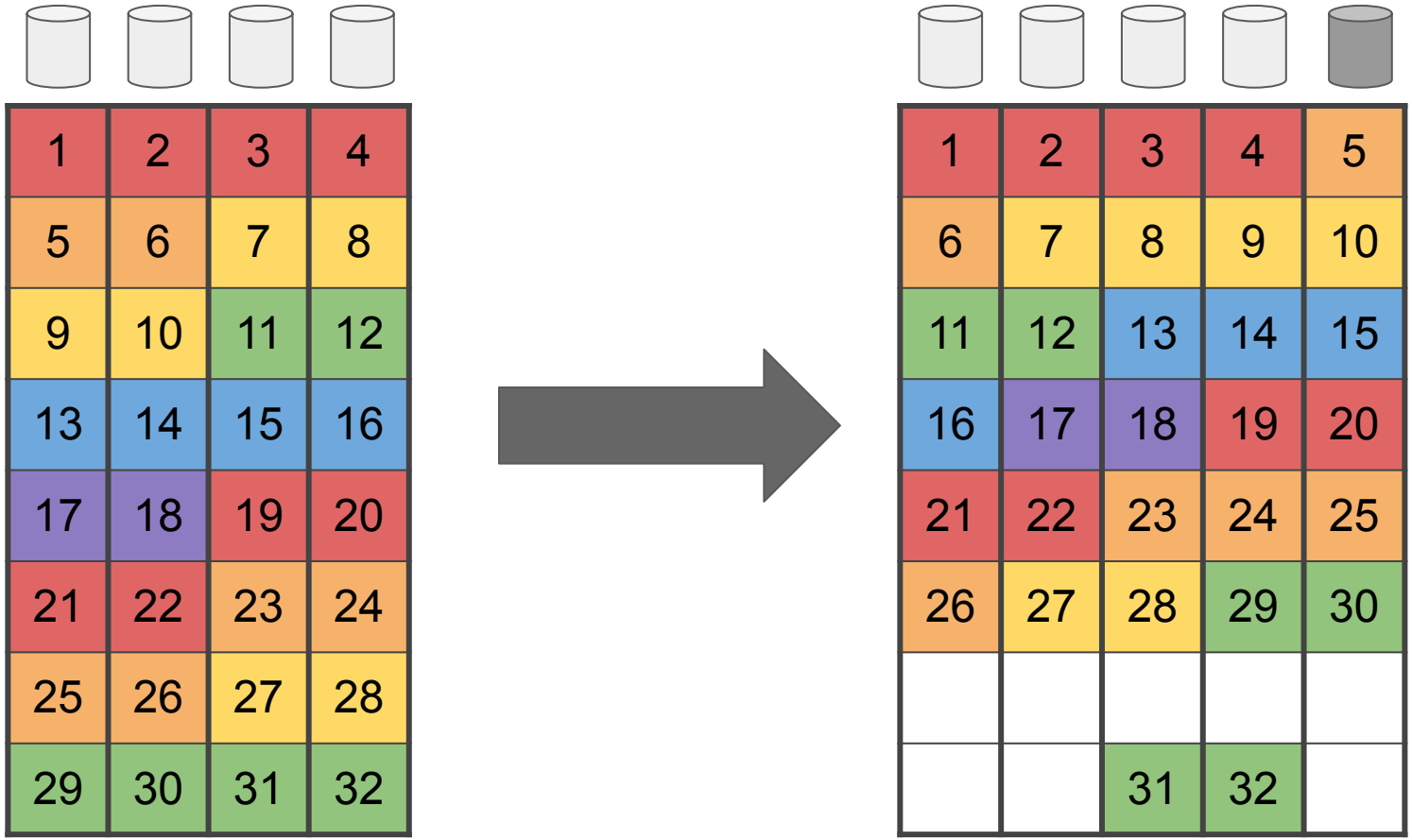
Reflow end state



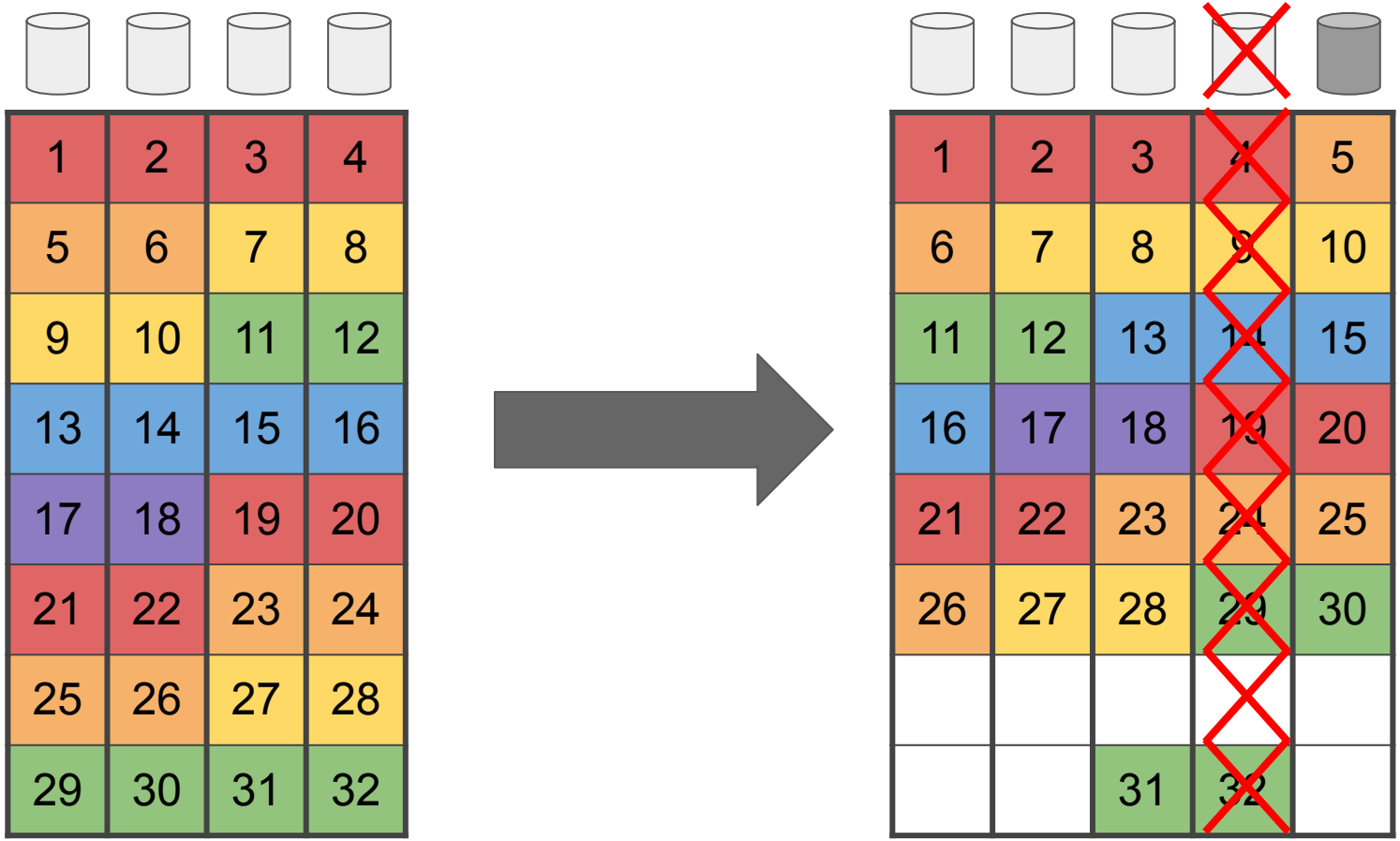
Reflow initial state



Possible intermediary state



Possible intermediary state: disk failure



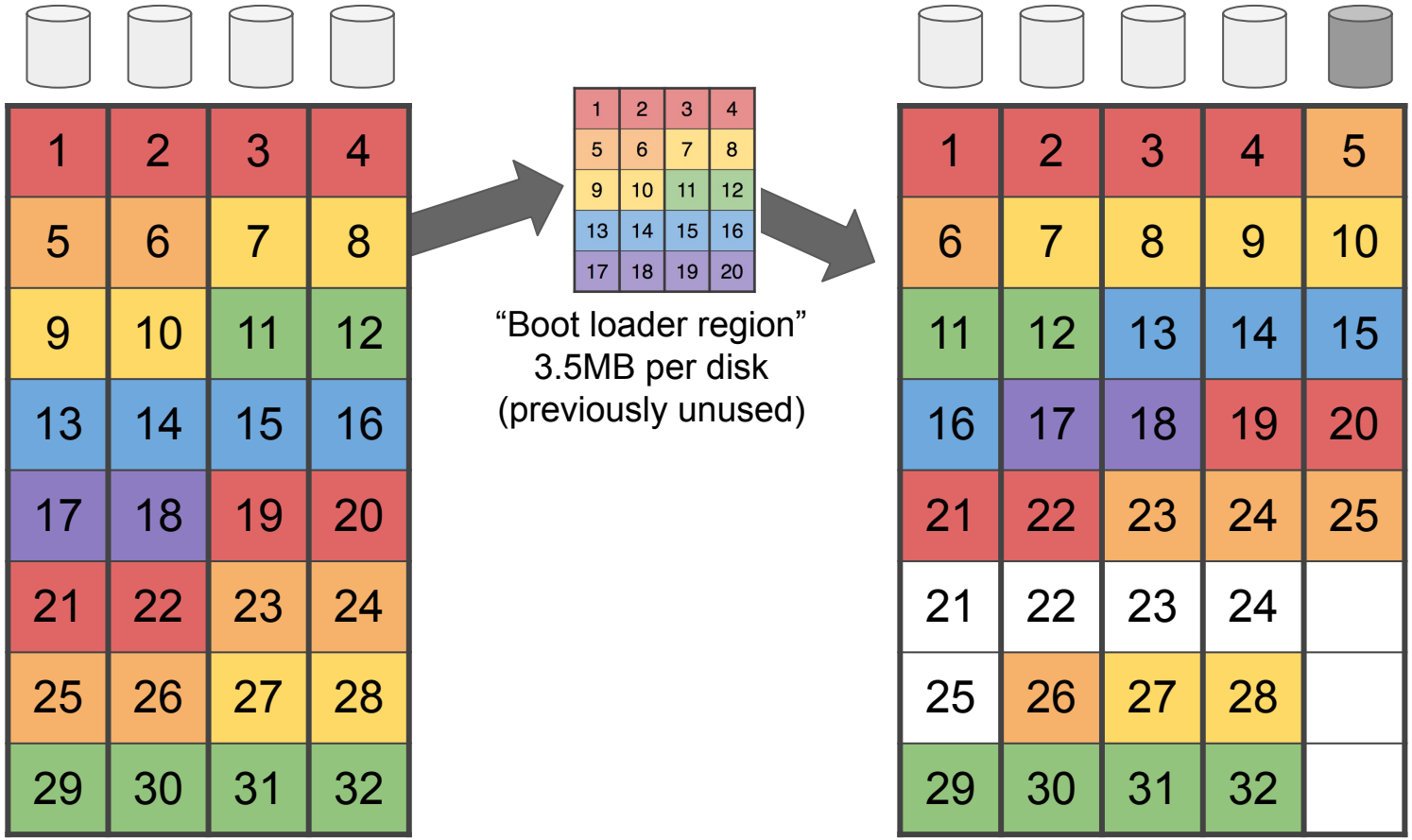
Reflow progress = 30
What if we lose a disk?
Read stripe @29-32

- If we read from new loc 29,30 & original loc 31,32
  - Data loss :-(
- Instead, read “split stripe” from original location
  - Same as if we hadn’t started moving this stripe

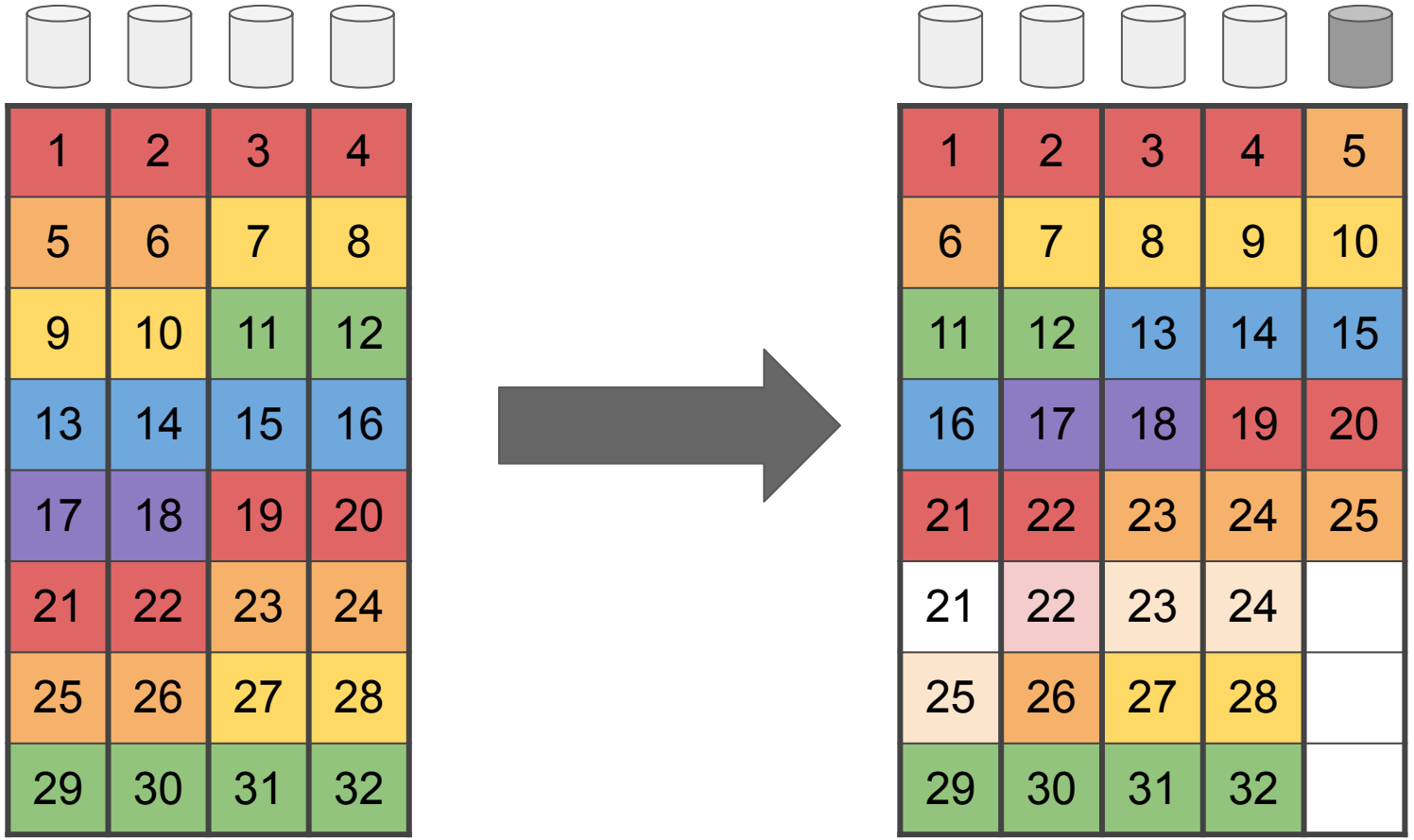
- **Need to ensure the original location of split stripe hasn’t been overwritten**

- Each stripe of block considered independently
- Only split **stripe** read from original loc (not whole block)

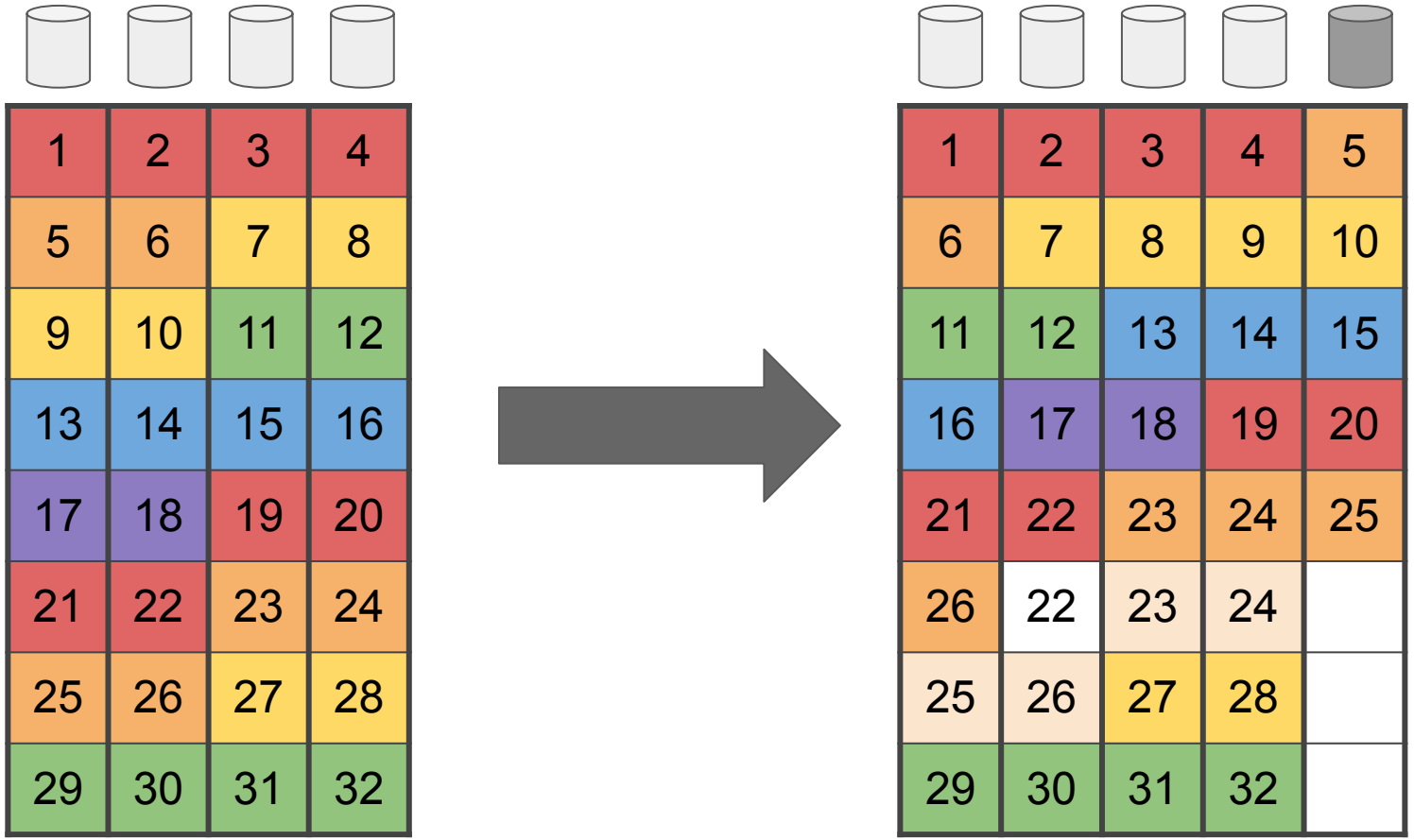
Read at least 5*5=25 sectors into scratch space



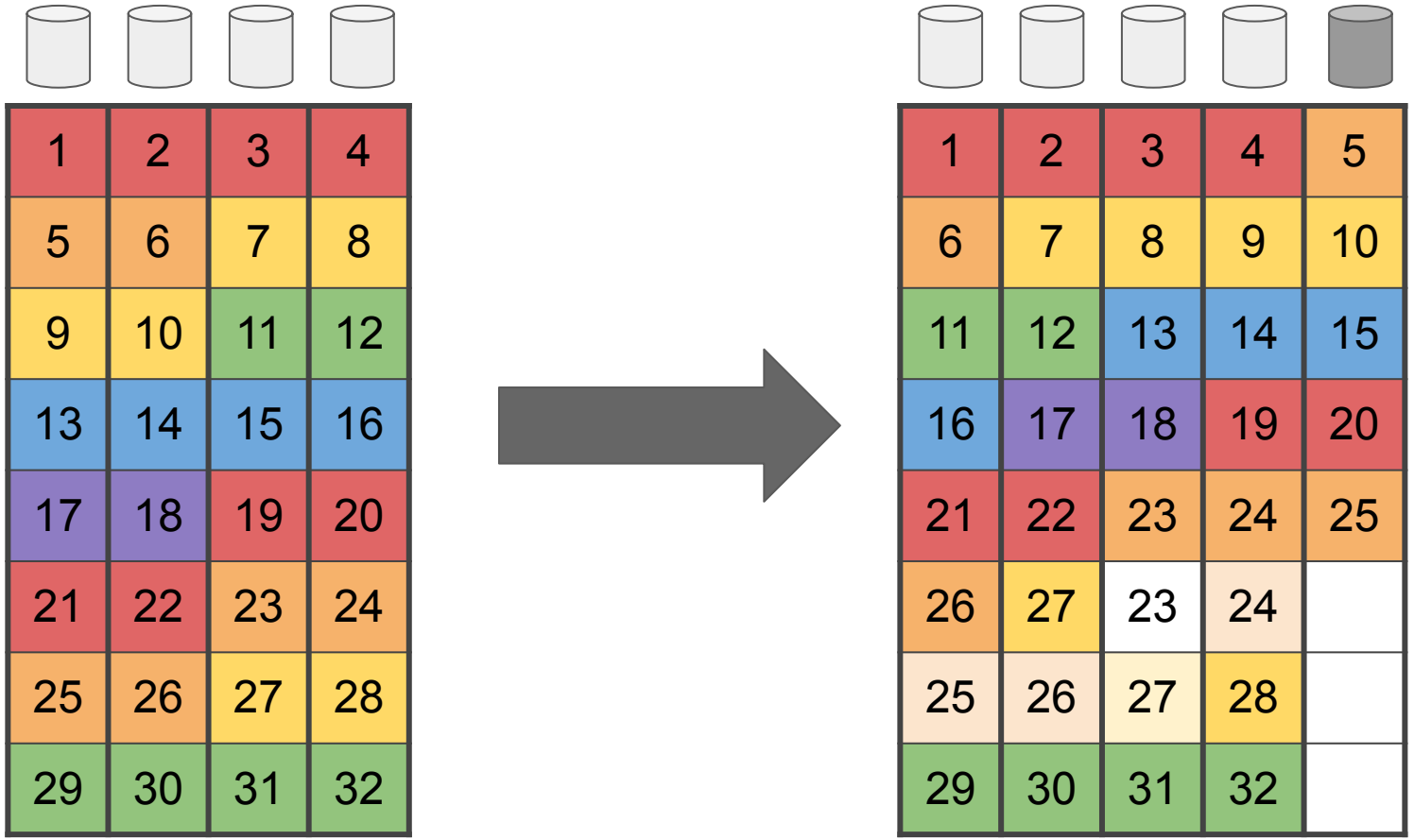
Reflow progress = 25; separation=6; chunk size = 1



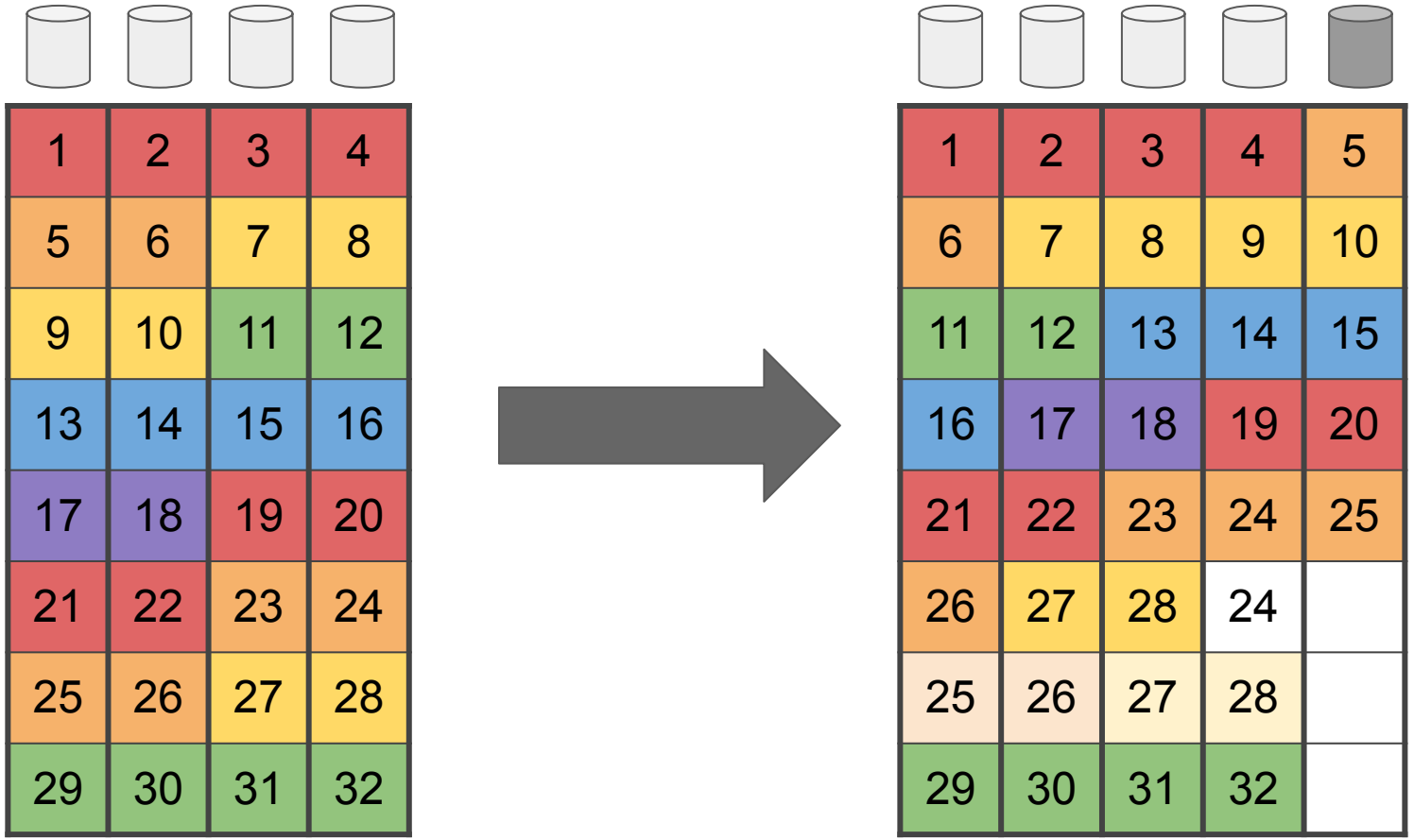
Reflow progress = 26; separation=6; chunk size = 1



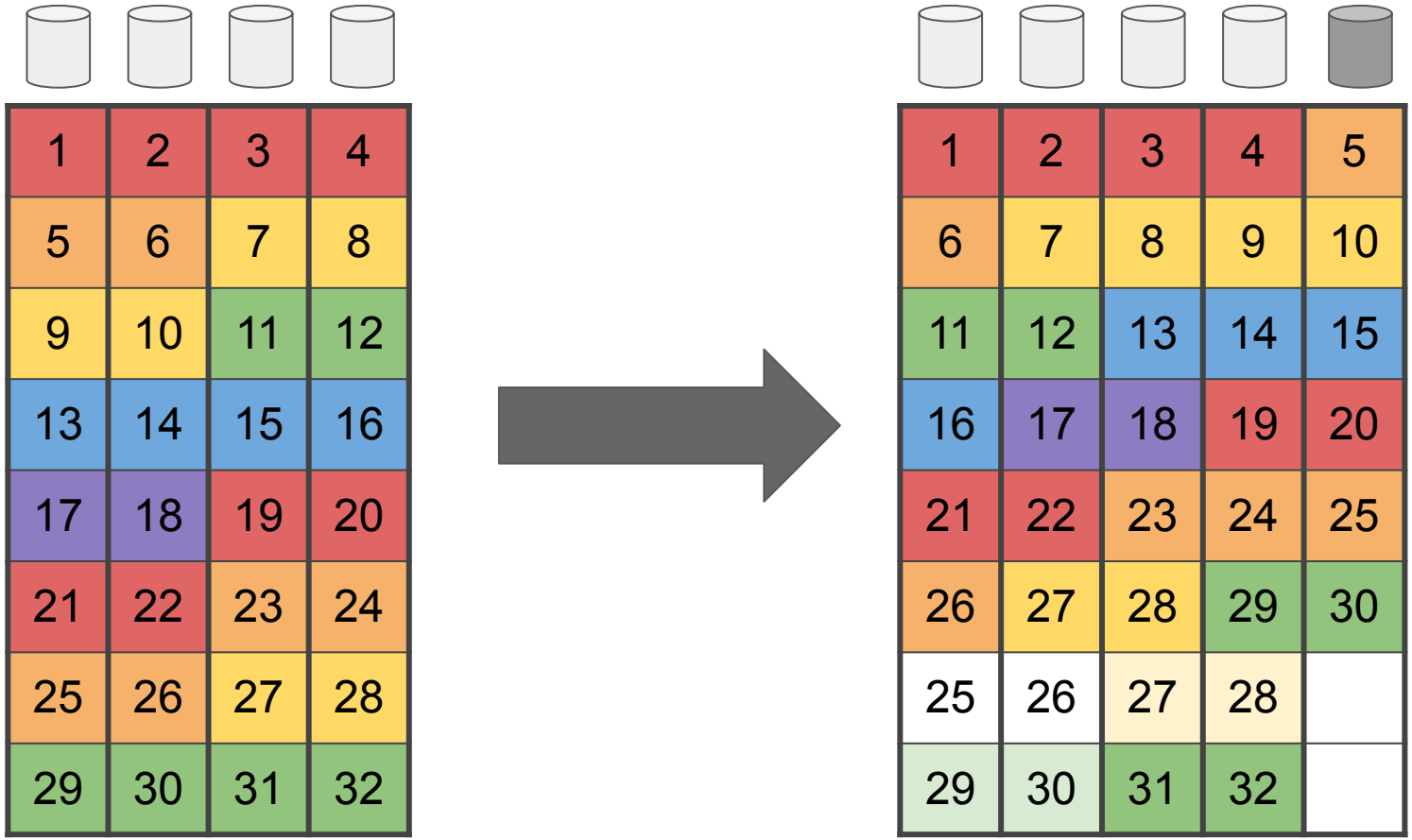
Reflow progress = 27; separation=6; chunk size = 1



Reflow progress = 28; separation=7; chunk size = 2



Reflow progress = 30; separation=7; chunk size = 2



Design implications

- Works with RAIDZ-1/2/3
- Can expand multiple times (4-wide -> 5 wide -> 6 wide)
- Old data has old Data : Parity ratio
- New data has new Data : Parity ratio
  - E.g. 4-wide RAIDZ1 -> 5-wide uses ~20% less space
- RAIDZ must be healthy (no missing devices) during reflow
  - If disk dies, reflow will pause and wait for it to be reconstructed

djb@mayqueen: ~ ⌥⌘5

```
djb@mayqueen:~$ uname -a
Linux mayqueen 6.2.0-34-generic #34-Ubuntu SMP PREEMPT_DYNAMIC Mon Sep  4 13:06:55 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
djb@mayqueen:~$ zp
```

Testing RAIDZ Expansion

- ZFS Test Suite Tests Added
  - Tested with all parity types
  - Multiple expansions
  - Offline/replace during expansion
  - Scrub and resilver - during and post expansion
  - Negative tests - feature disabled, checkpoints, scratch in-use, etc.
  - Uses `expand_max_reflow_bytes` testing tunable
- New ztest testing options
  - --raidz-expansion – dedicated mode, focused on crashing during expansion
  - --raid-kind=raidz – used with standard mix-it-up mode
- Updated zloop to drive RAIDZ Expansion configurations/options
  - P1: 3 - 7 disks, P2: 5 - 9 disks, P3: 7 - 11 disks
  - Picks one of the above ztest mode options
- Manual functional testing on real hardware (sata HDDs)
- Manual performance testing (before/during/after expansion)

Example Test Runs

raidz expansion, 11 data, 3 parity, ashift 9

```
$ scripts/zloop.sh
Setting core file pattern...
10/13 21:15:30 /home/zfs/ztest -G -VVVVV -K eraidz -m 0 -r 11 -D 0 -S 0 -R 3 -v 1 -a 9 -C special=random -s 512m
10/13 21:21:04 /home/zfs/ztest -G -VVVVV -K raidz -m 0 -r 7 -D 0 -S 0 -R 2 -v 5 -a 12 -C special=random -s 512m
10/13 21:22:50 /home/zfs/ztest -G -VVVVV -K raidz -m 1 -r 4 -D 0 -S 0 -R 2 -v 4 -a 12 -C special=random -s 512m
10/13 21:24:23 /home/zfs/ztest -G -VVVVV -K raidz -m 2 -r 0 -D 0 -S 0 -R 1 -v 2 -a 9 -C special=random -s 512m
10/13 21:30:01 /home/zfs/ztest -G -VVVVV -K draid -m 0 -r 11 -D 4 -S 4 -R 3 -v 0 -a 9 -C special=random -s 512m
10/13 21:35:33 /home/zfs/ztest -G -VVVVV -K eraidz -m 0 -r 6 -D 0 -S 0 -R 1 -v 1 -a 12 -C special=random -s 512m
10/13 21:41:26 /home/zfs/ztest -G -VVVVV -K raidz -m 0 -r 10 -D 0 -S 0 -R 1 -v 5 -a 12 -C special=random -s 512m
10/13 21:41:57 /home/zfs/ztest -G -VVVVV -K raidz -m 2 -r 0 -D 0 -S 0 -R 1 -v 2 -a 12 -C special=random -s 512m
10/13 21:48:00 /home/zfs/ztest -G -VVVVV -K raidz -m 2 -r 5 -D 0 -S 0 -R 3 -v 4 -a 9 -C special=random -s 512m
10/13 21:49:47 /home/zfs/ztest -G -VVVVV -K draid -m 0 -r 50 -D 8 -S 3 -R 3 -v 1 -a 9 -C special=random -s 512m

$ scripts/zfs-tests.sh -T raidz
Test: /home/zfs/tests/zfs-tests/tests/functional/raidz/raidz_expand_001_pos (run as root) [00:42] [PASS]
Test: /home/zfs/tests/zfs-tests/tests/functional/raidz/raidz_expand_002_pos (run as root) [01:10] [PASS]
Test: /home/zfs/tests/zfs-tests/tests/functional/raidz/raidz_expand_003_neg (run as root) [00:10] [PASS]
Test: /home/zfs/tests/zfs-tests/tests/functional/raidz/raidz_expand_003_pos (run as root) [00:35] [PASS]
Test: /home/zfs/tests/zfs-tests/tests/functional/raidz/raidz_expand_004_pos (run as root) [02:03] [PASS]
Test: /home/zfs/tests/zfs-tests/tests/functional/raidz/raidz_expand_005_pos (run as root) [01:28] [PASS]
Test: /home/zfs/tests/zfs-tests/tests/functional/raidz/raidz_expand_006_neg (run as root) [00:00] [PASS]
Test: /home/zfs/tests/zfs-tests/tests/functional/raidz/raidz_expand_007_neg (run as root) [00:00] [PASS]
```


performance (ashift=9, 3->4-wide RAIDZ1)
Worst case
write pre-expansion
1800
read pre-expansion
2600
expansion
50
Huge CPU costs
of 512-byte zio's
(~6 hours/TB)
Less parity
50%->33%
write post-expansion
2000
Have to Read parity
(+50% bytes)
read post-expansion
2300
1000
2000
3000
Logical MB/s (file writes)

Status (final 🙏 report)

- PR under review

https://github.com/openzfs/zfs/pull/15022

- Future work
  - Performance of reflow
  - Add multiple drives at once?
  - Rewrite all existing blocks with new data:parity ratio?
  - DRAID expansion???
  - Auto scrub before and/or after?

- Not part of this design
  - Add parity (RAIDZ1 -> RAIDZ2)
  - Remove disks
  - Defragment
  - Mirror -> RAIDZ

Thank you!

klara

Fedor Uporov (vStack)
Stuart Maybee (FreeBSD Foundation)