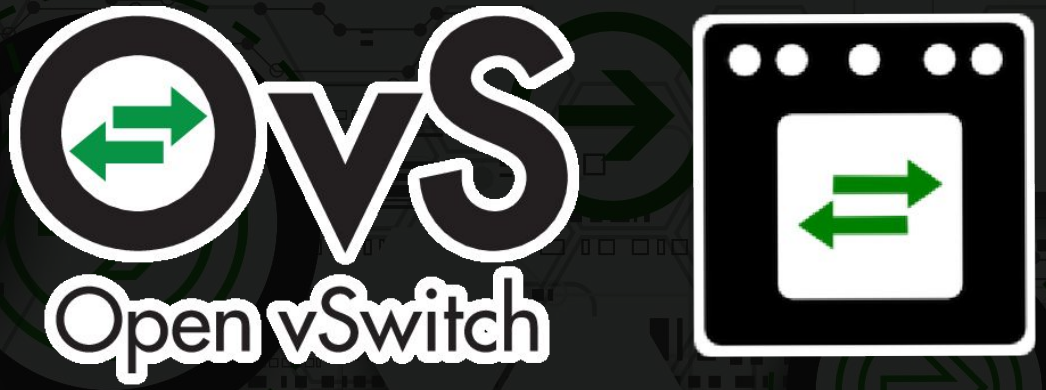


Prague, November 19-20, 2025

Improving Retrieval from the OVS Hashmap

Rosemarie O'Riorden, Red Hat

Agenda

1. Why?
2. OVS's current hashmap
3. The improved hashmap design
4. Pros and cons
5. Results & next steps

Why?

- sset, smap
- options/config

OVS:

- Datapath ports
- Port name mappings

OVN:

- Logical flows
- Datapaths

Why?

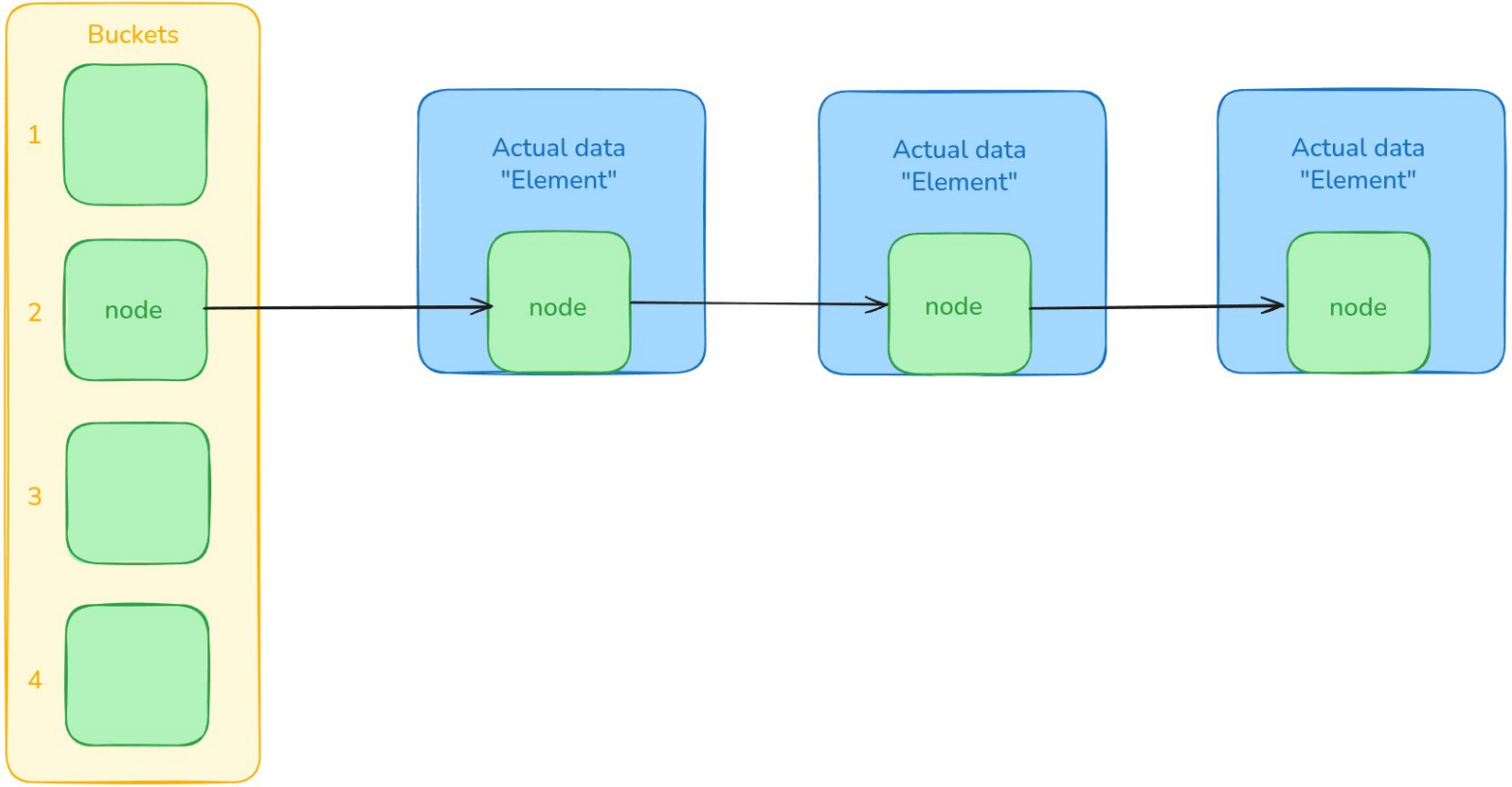
```
~/ovs $ git grep HMAP_FOR_EACH | wc -l
578

~/ovs $ git grep -i hmap | wc -l
2400

~/ovn $ git grep HMAP_FOR_EACH | wc -l
453

~/ovn $ git grep -i hmap | wc -l
3036
```

OVS's Current Hashmap



The Idea

A new hash table

2025-03-28 · *Viktor Söderqvist*

The Idea

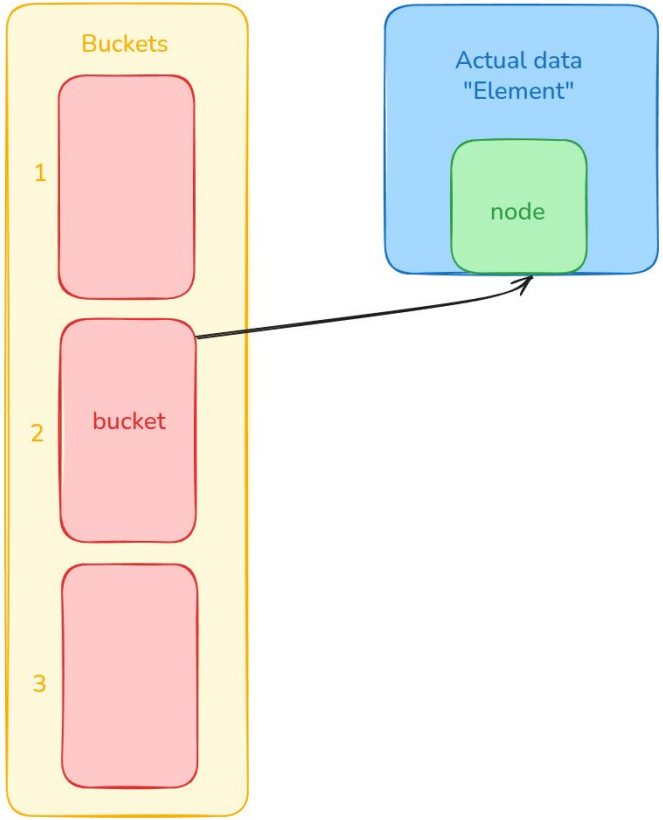
A new hash table

2025-03-28 · *Viktor Söderqvist*

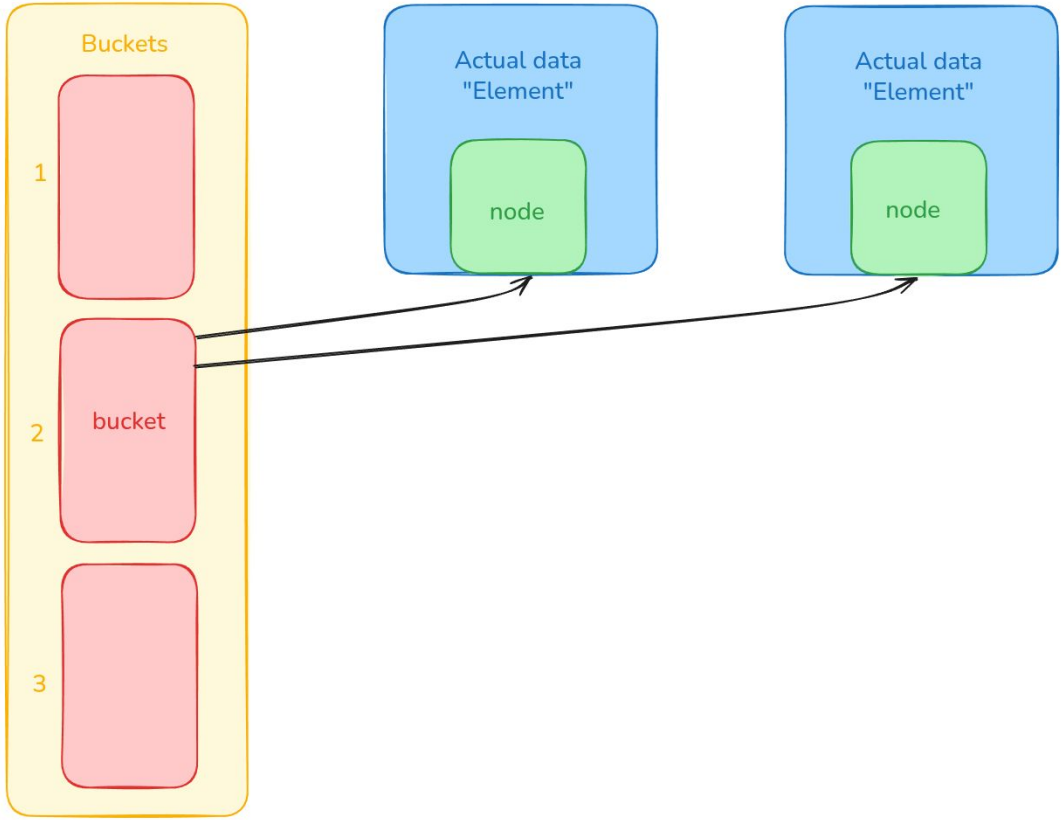
Valkey achieved:

20 bytes less memory used per element

The Improved Hashmap

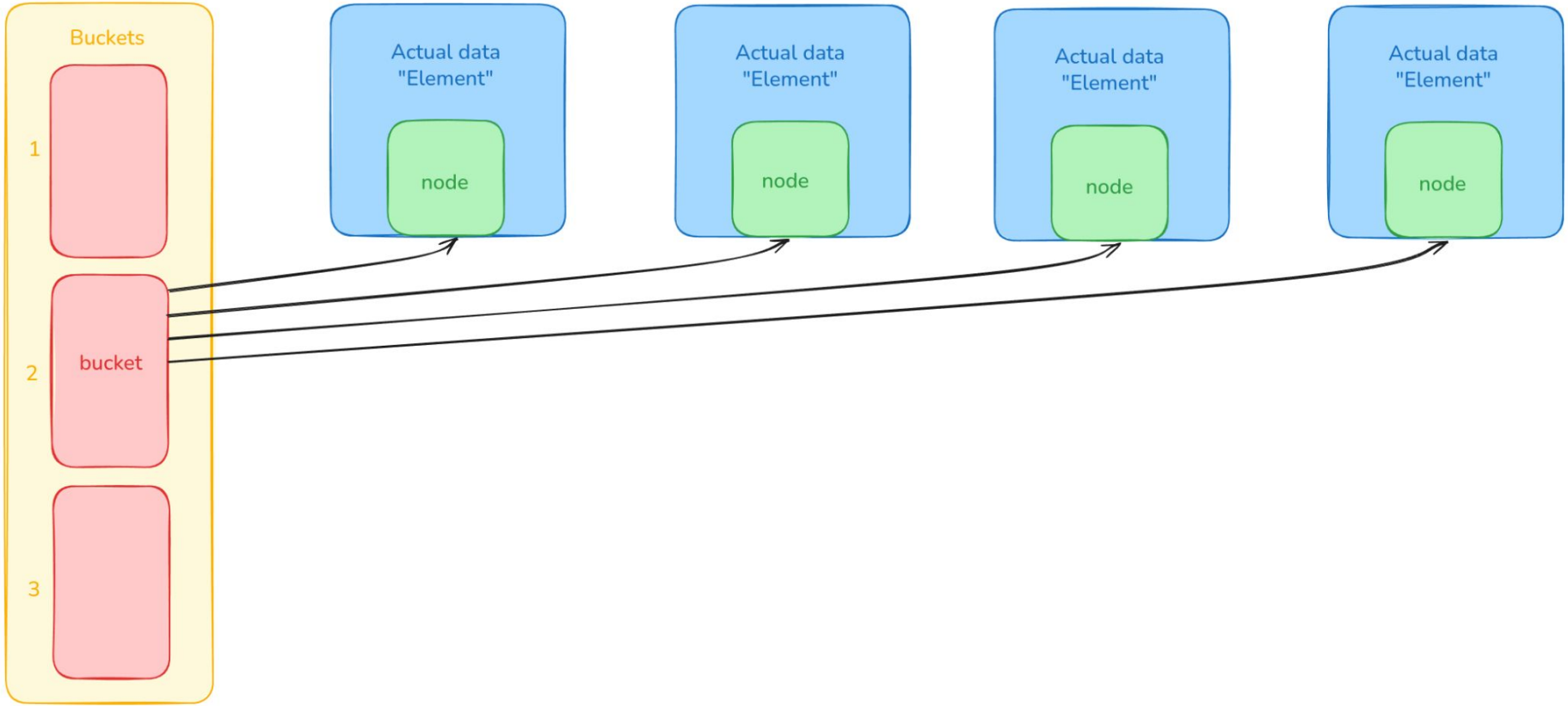


The Improved Hashmap

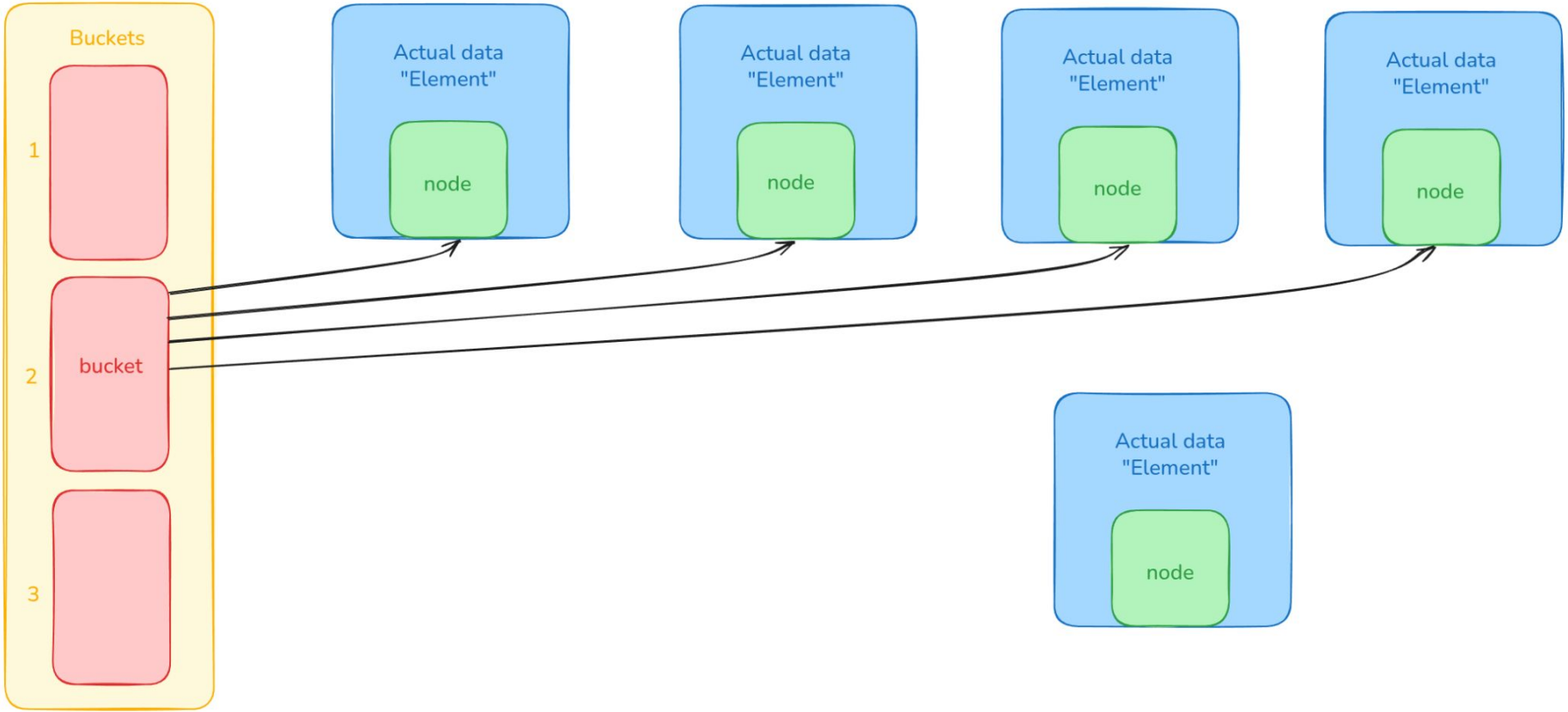


The Improved Hashmap

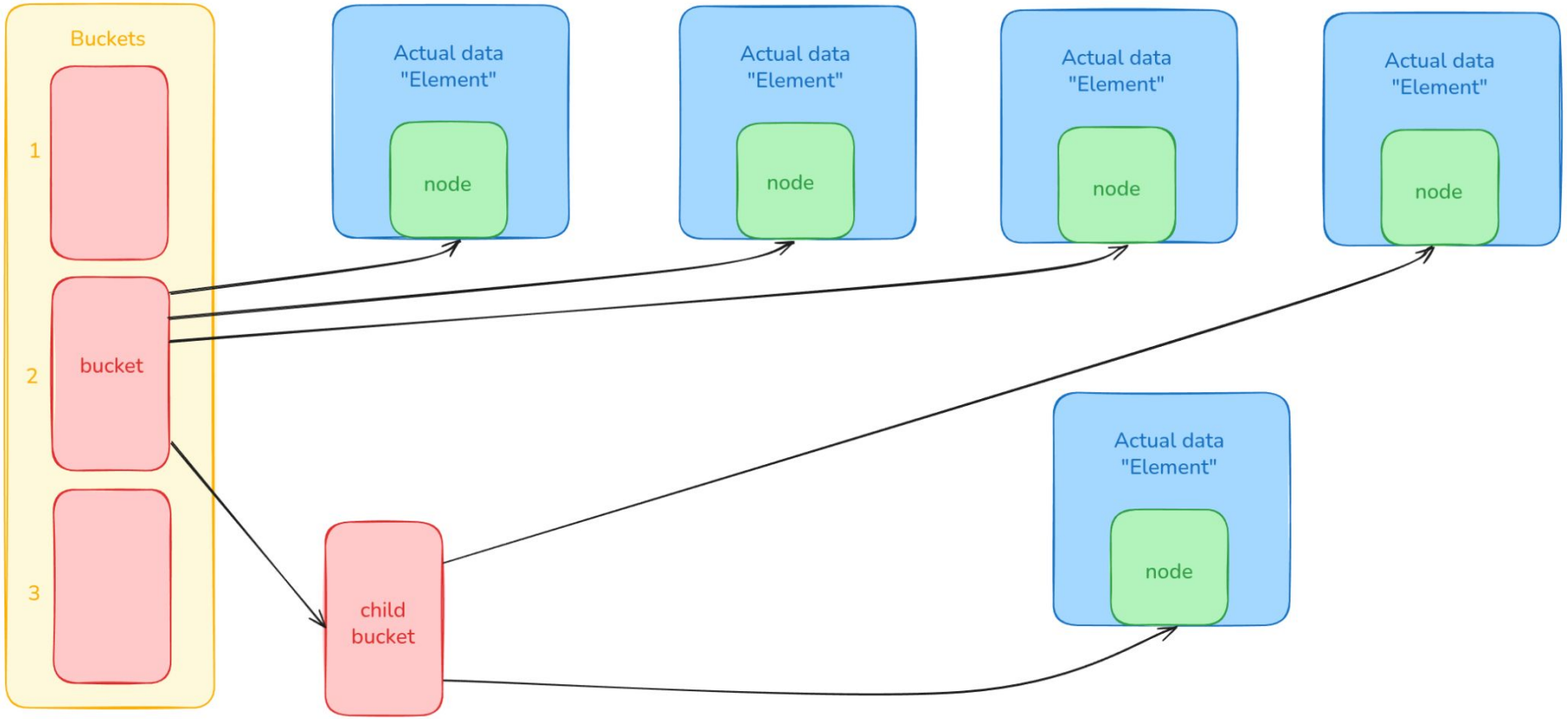
The Improved Hashmap



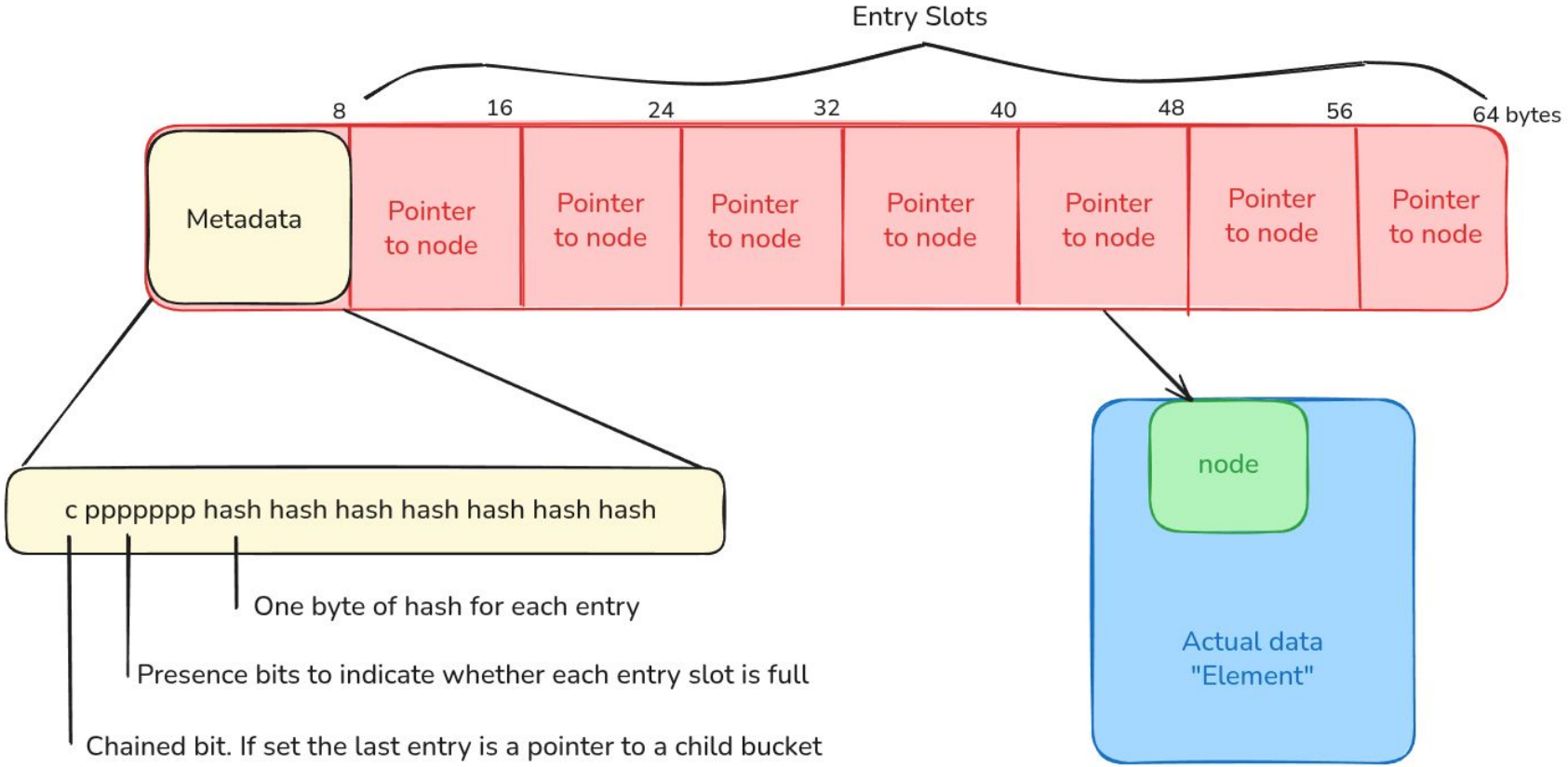
The Improved Hashmap



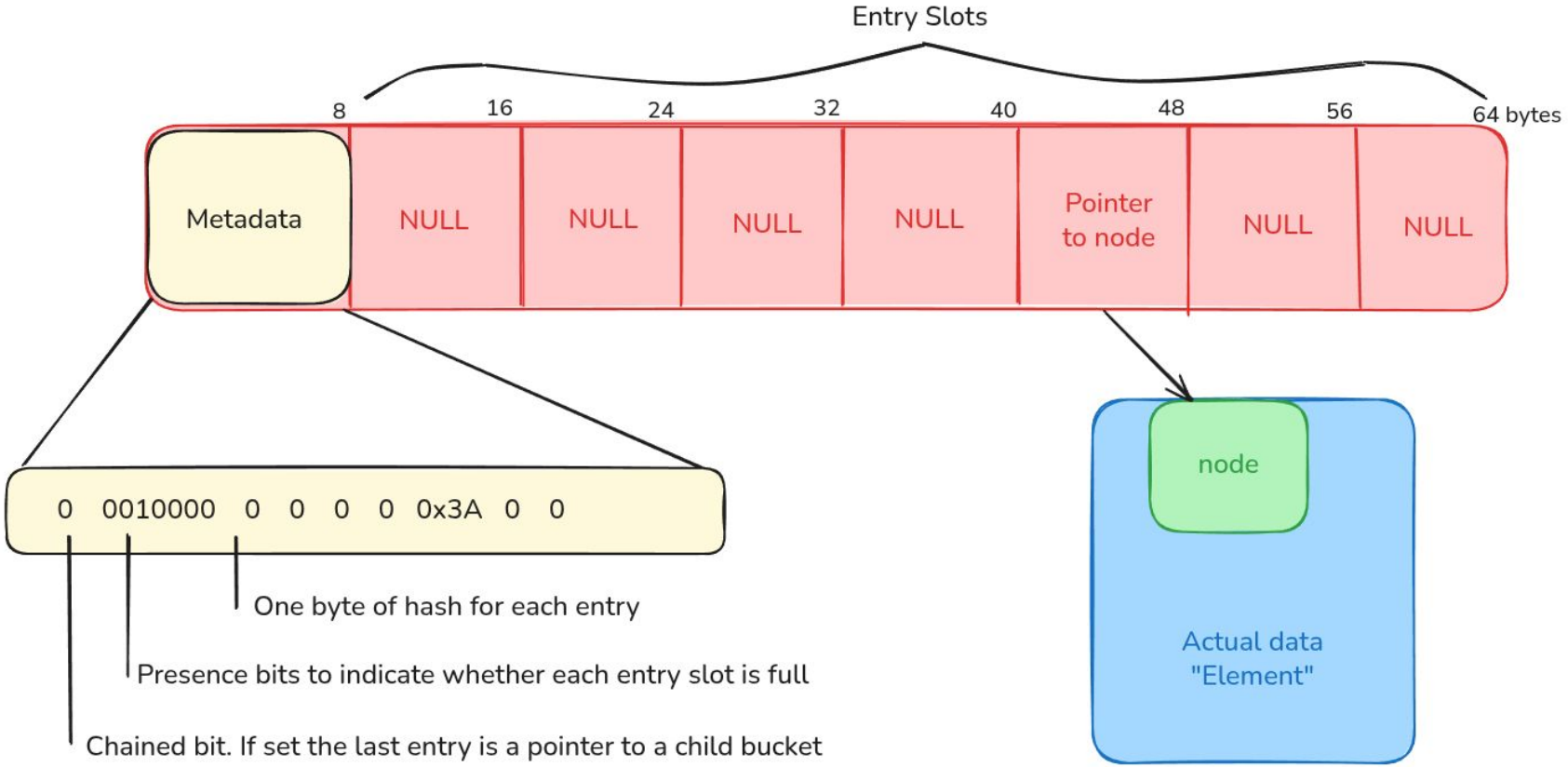
The Improved Hashmap



Bucket details



Bucket details (example)



Perceived Pros and Cons

Pros:

- Saves space in cache
- Less cache misses
- No need to loop through each node to remove
- <1% chance of hash-byte collision (cache miss)

Cons:

- More complex
- Less readable

Results

~50% slower...

Current hmap:

```
Benchmarking with n=10000000, 1 threads,
50.00% mutations

hmap insert:     826 ms
hmap iterate:   1246 ms
hmap search:     418 ms
hmap destroy:    632 ms
```

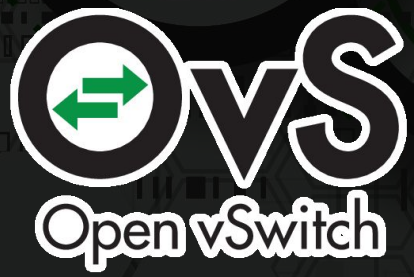
Proposed hmap:

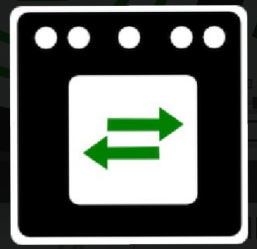
```
Benchmarking with n=10000000, 1 threads,
50.00% mutations

hmap insert:    2309 ms
hmap iterate:   1345 ms
hmap search:     618 ms
hmap destroy:    748 ms
```

Next steps

- Optimize
  - Avoid running child bucket logic as much as possible (especially in insert)
  - Test if the last pointer always being to a child bucket is more efficient





Thank you!

Questions?

Contact: rosemarie@redhat.com