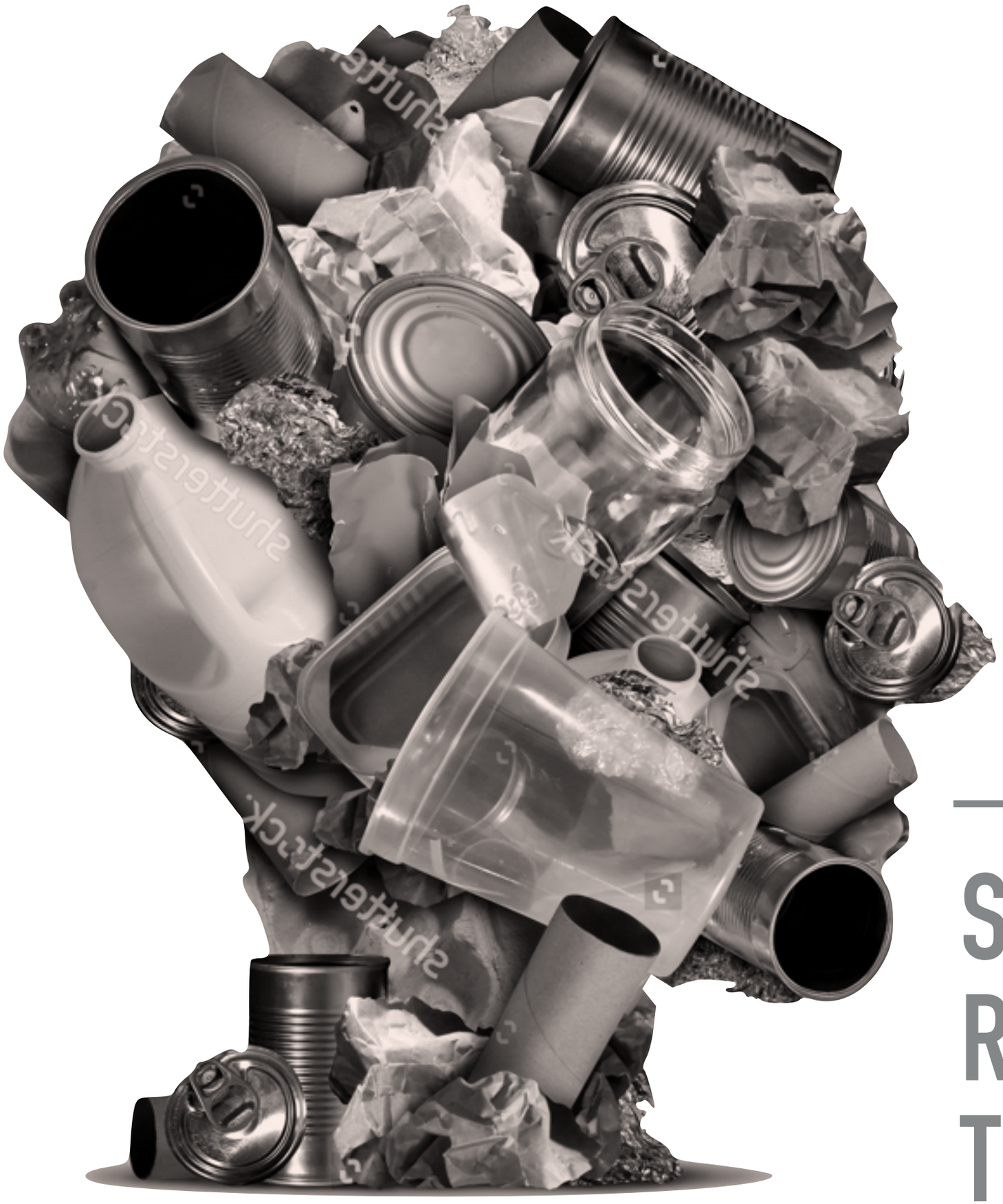


GARBAGE COLLECTION

SAURABH KADEKODI
RAJAT KATEJA
TIANYUAN DING

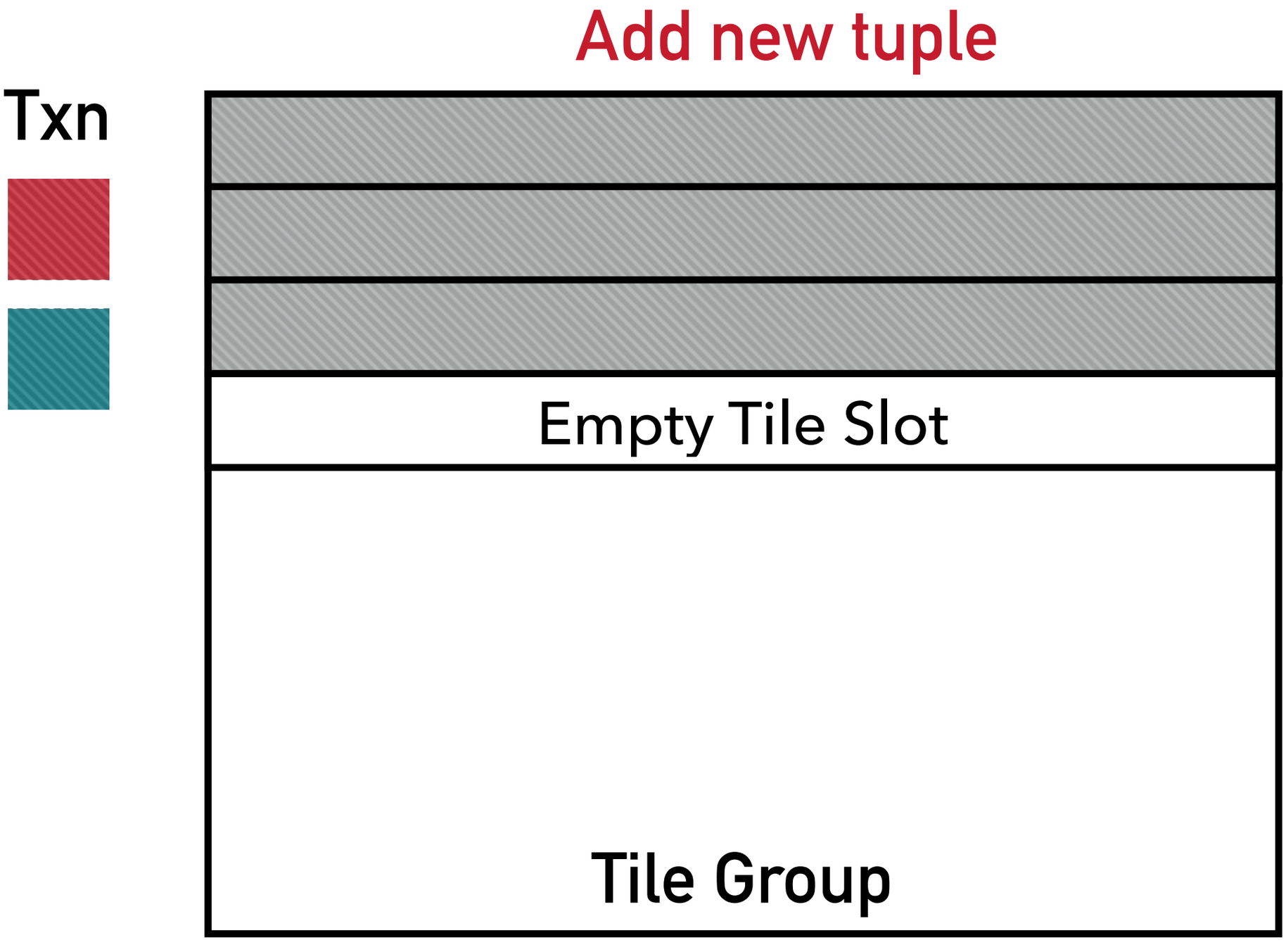
PROBLEM STATEMENT

- To implement efficient garbage collection in Peloton

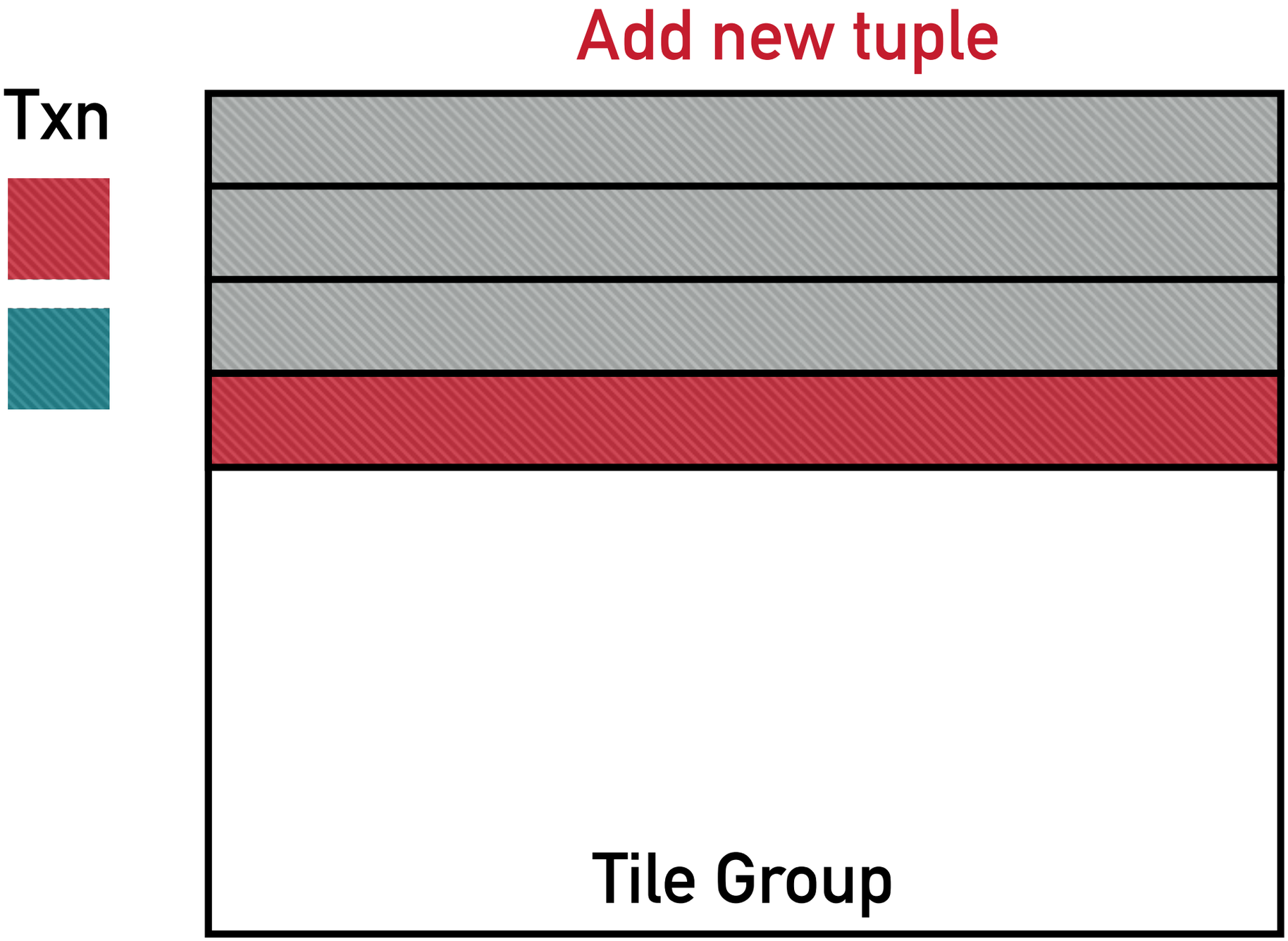
GOALS (REVISED)

- 75% - implement basic tuple recycling using vacuum
- 100% - implement epoch based co-operative gc
- 112.5% - lock-free implementations, GetMemoryFootprint()
- 125% - DDL garbage collection (deferred after discussion)

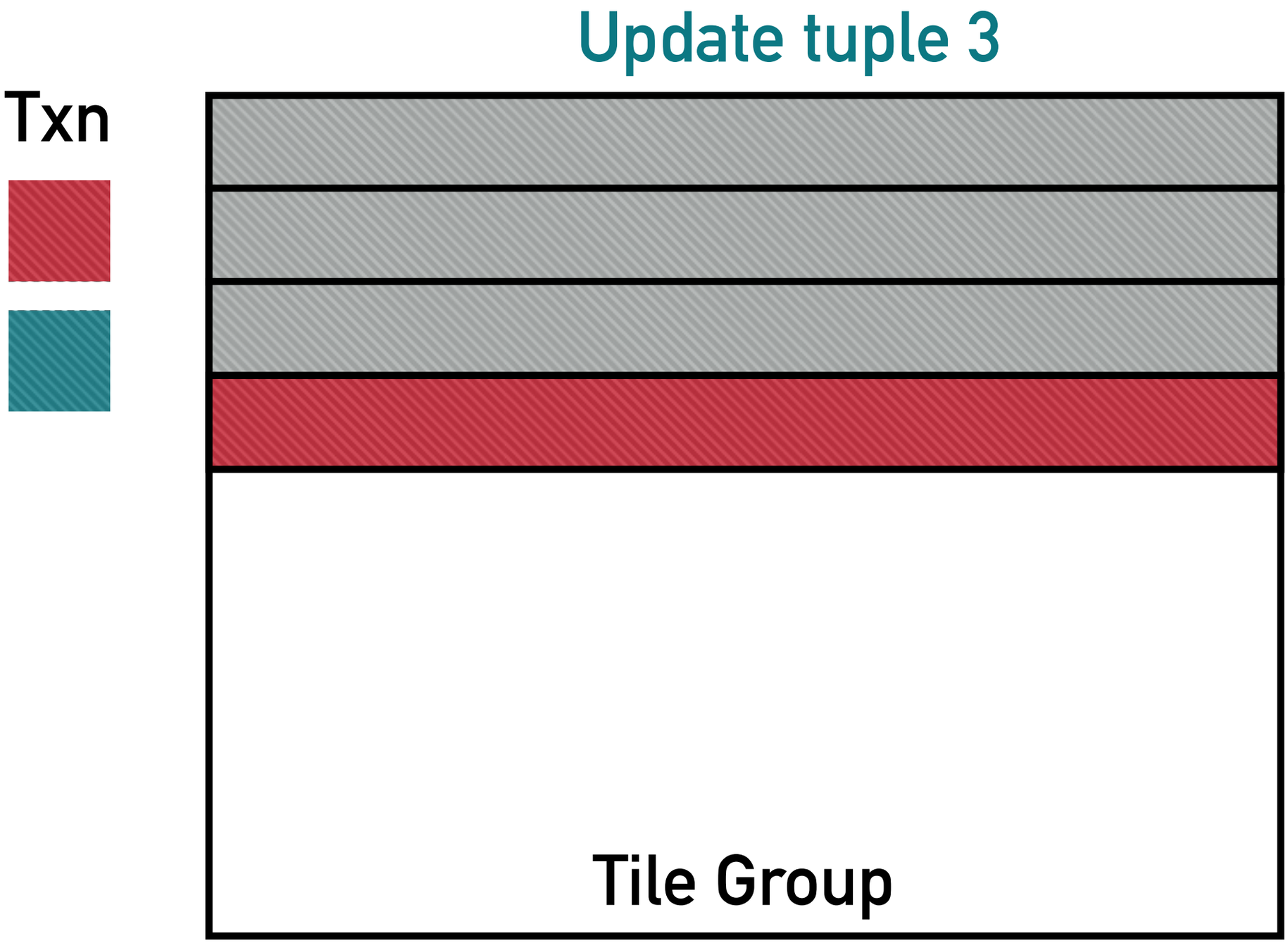
WITHOUT GC



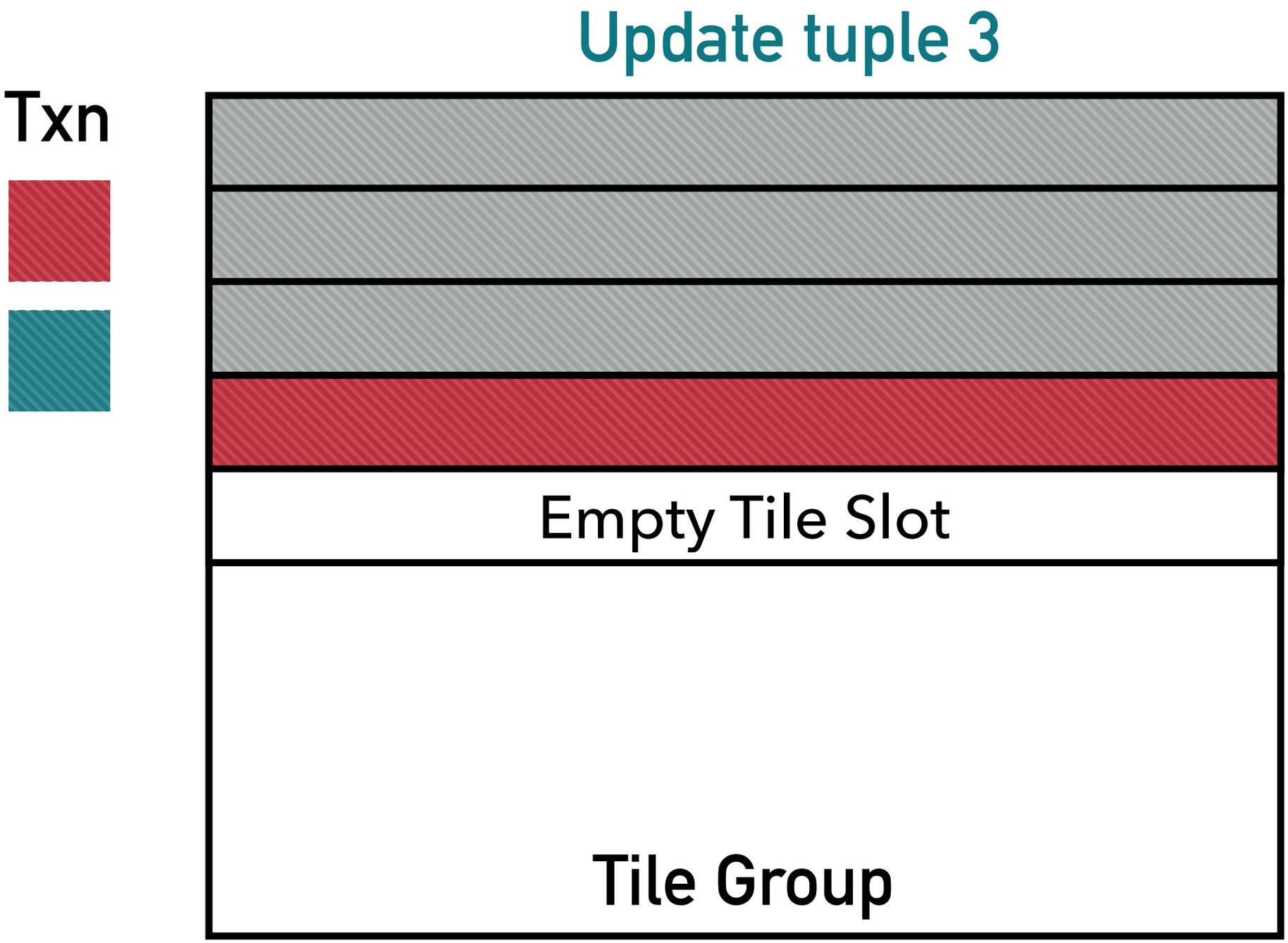
WITHOUT GC



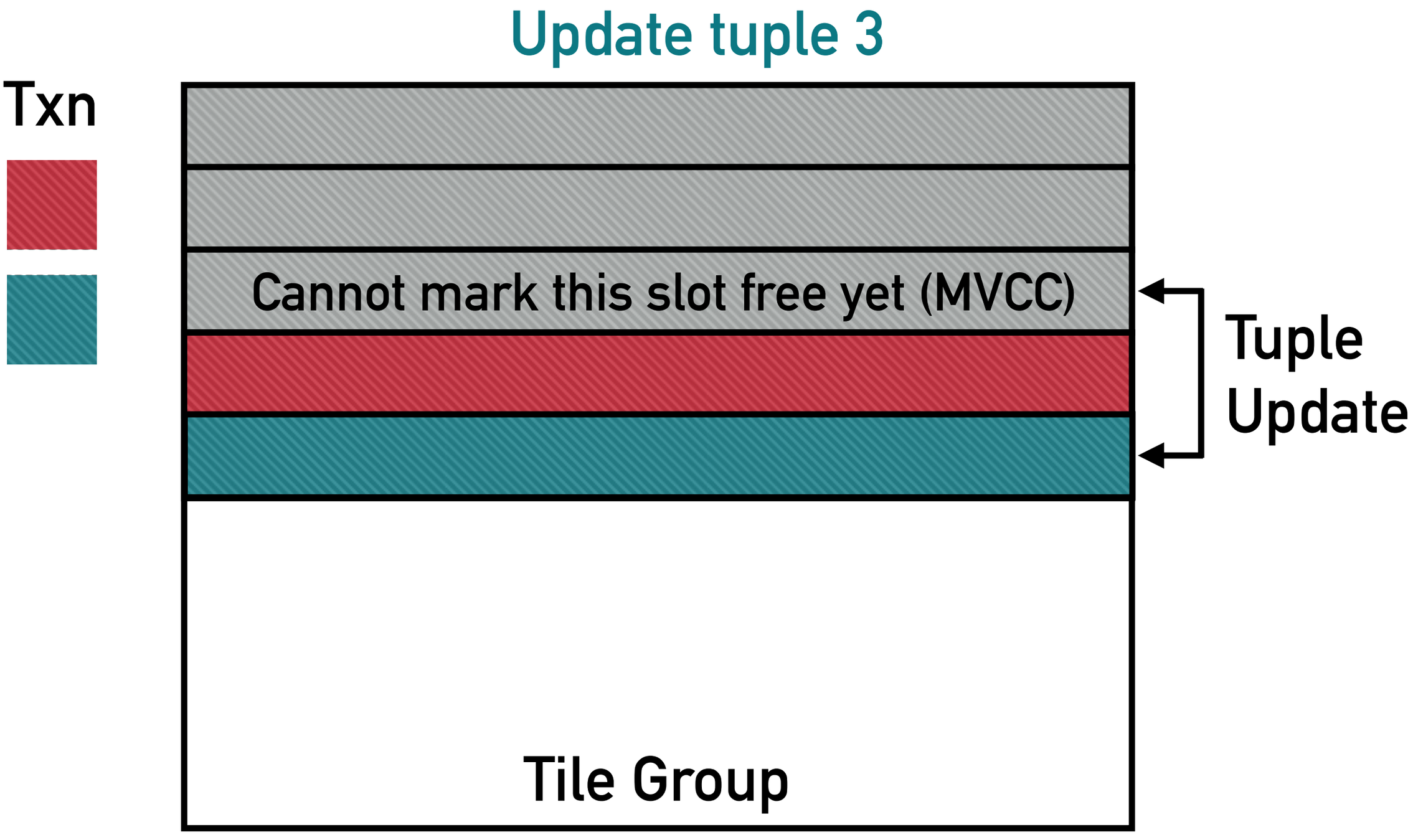
WITHOUT GC



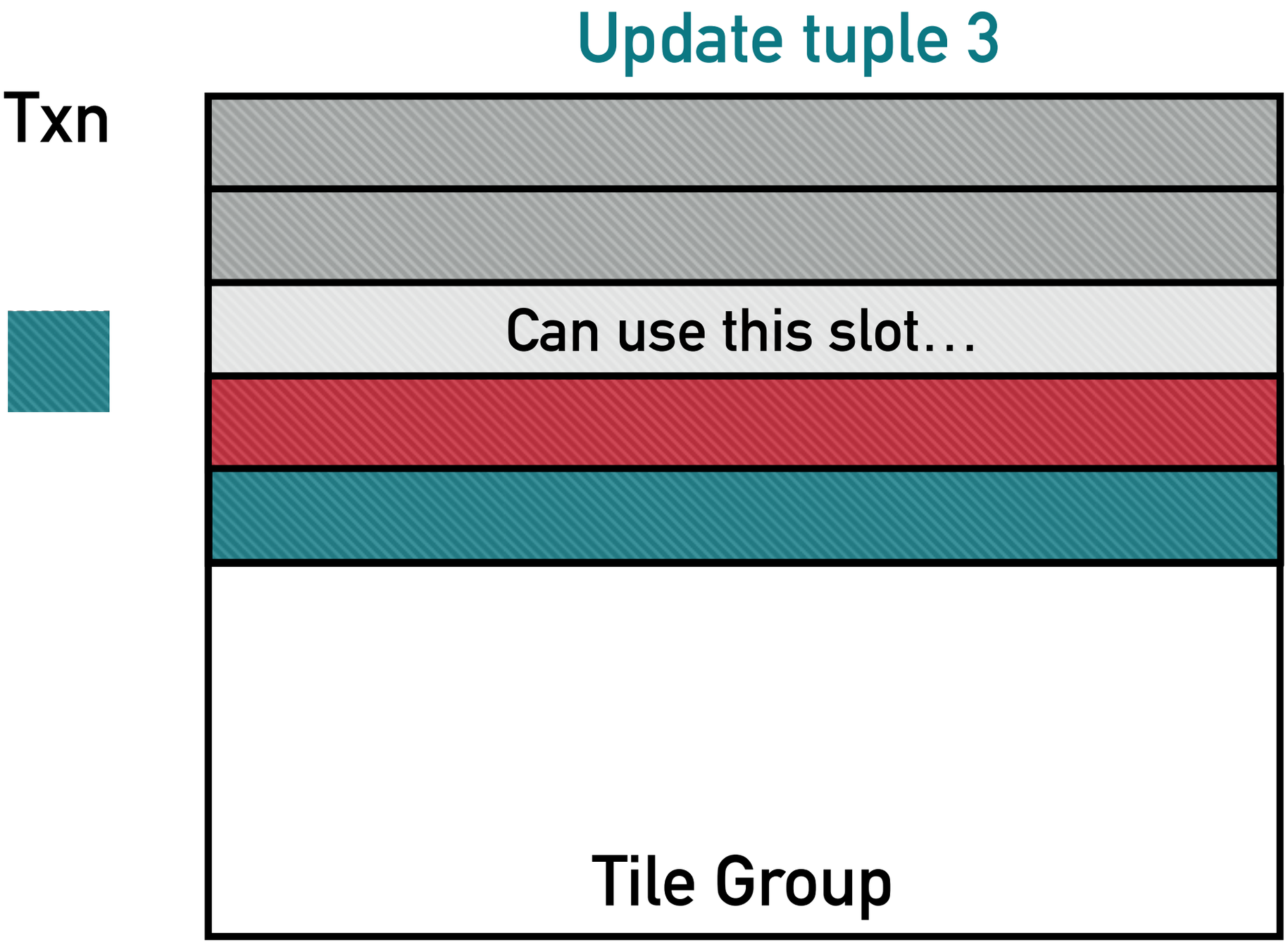
WITHOUT GC



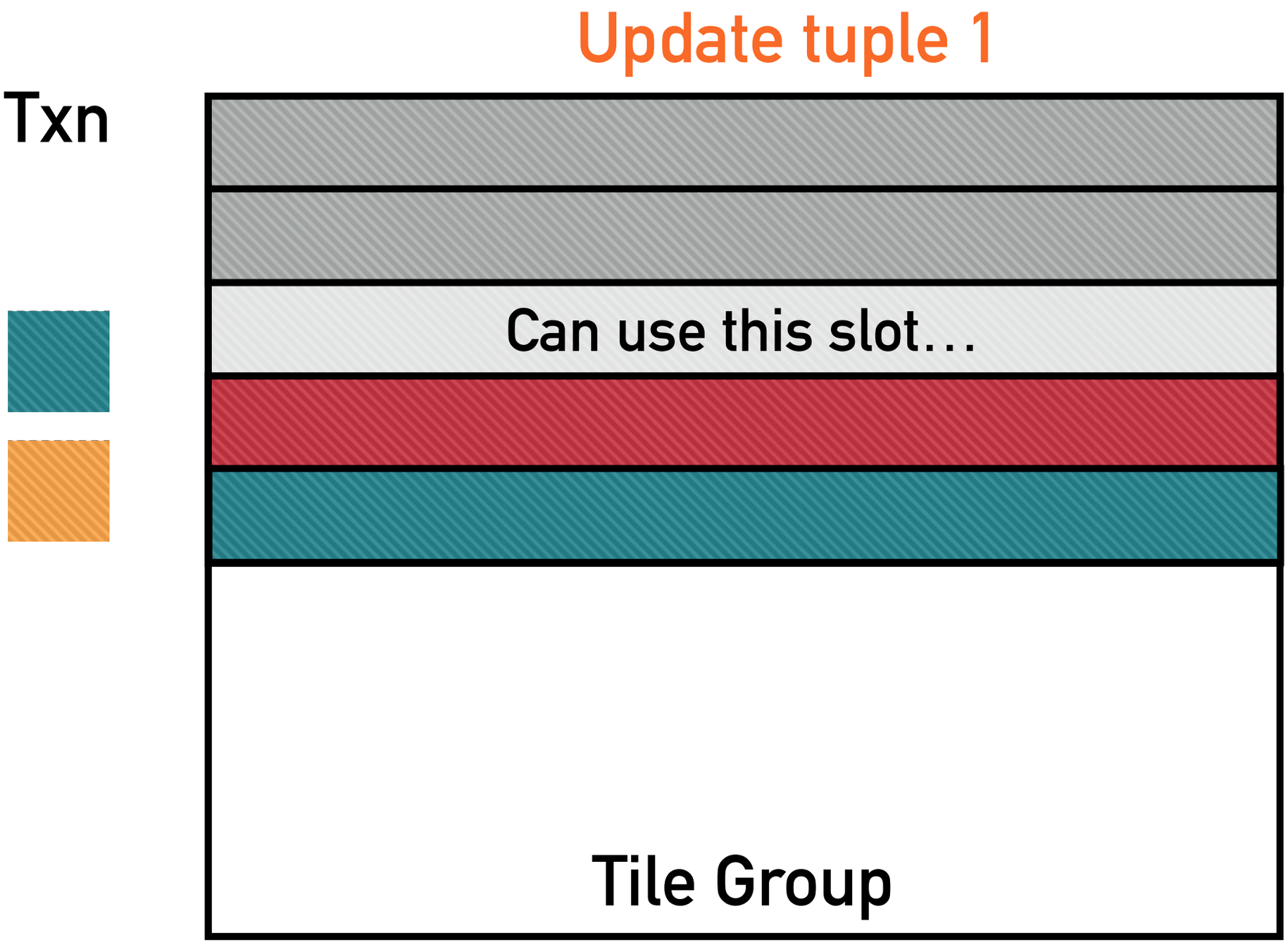
WITHOUT GC


Update tuple 3
Txn
Cannot mark this slot free yet (MVCC)
Tuple Update
Tile Group

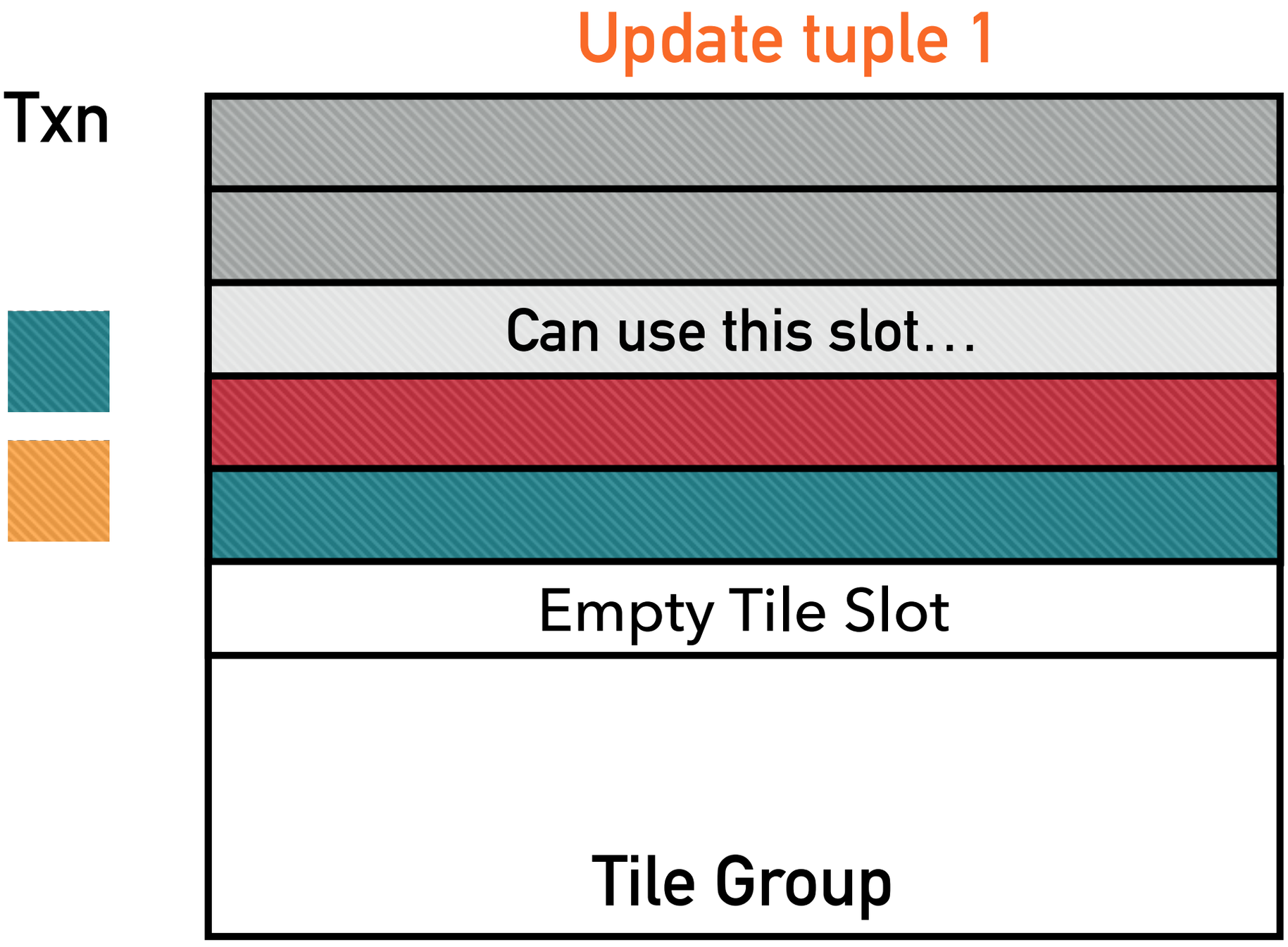
WITHOUT GC



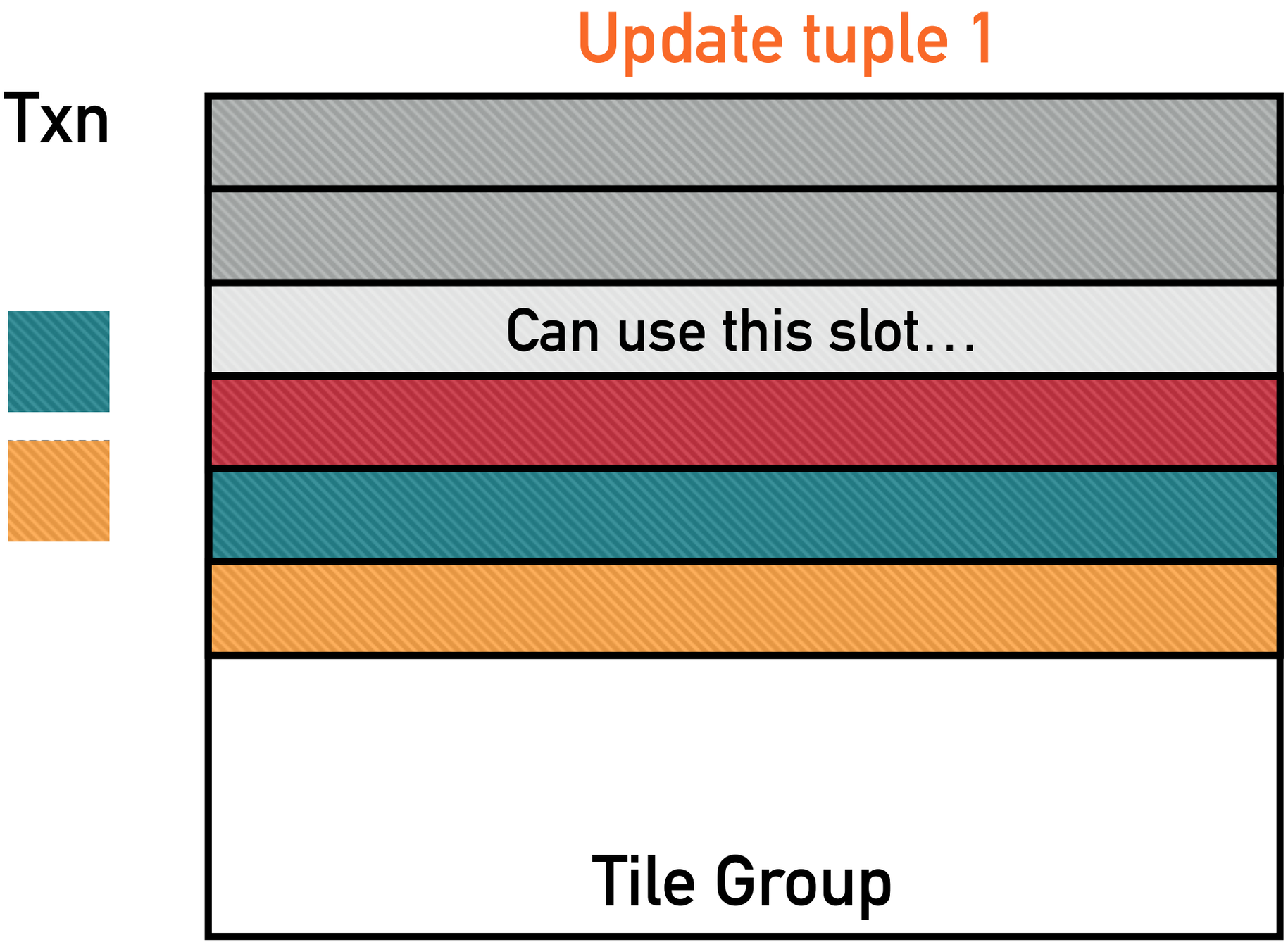
WITHOUT GC



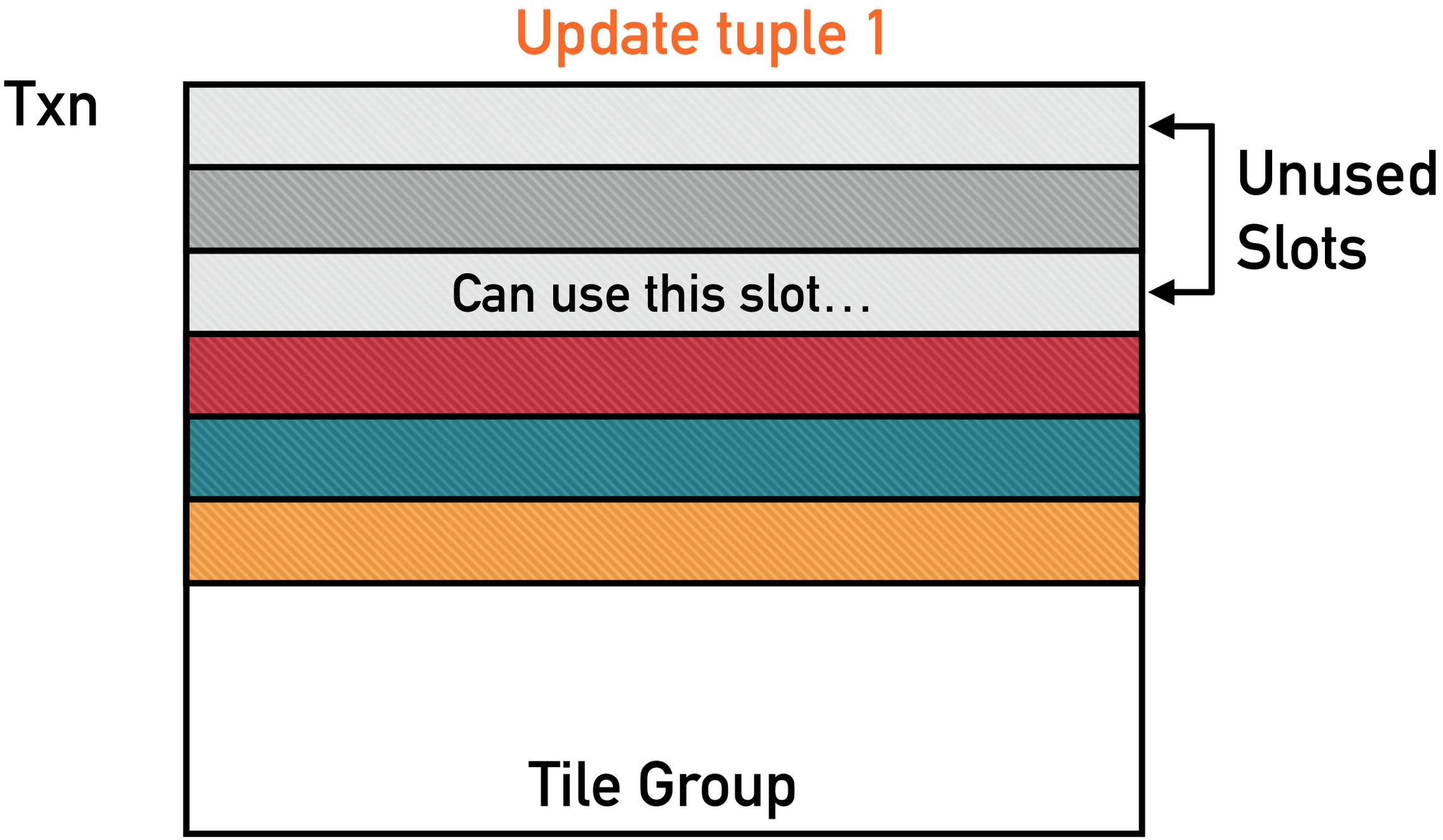
WITHOUT GC


Update tuple 1
Txn
Can use this slot…
Empty Tile Slot
Tile Group

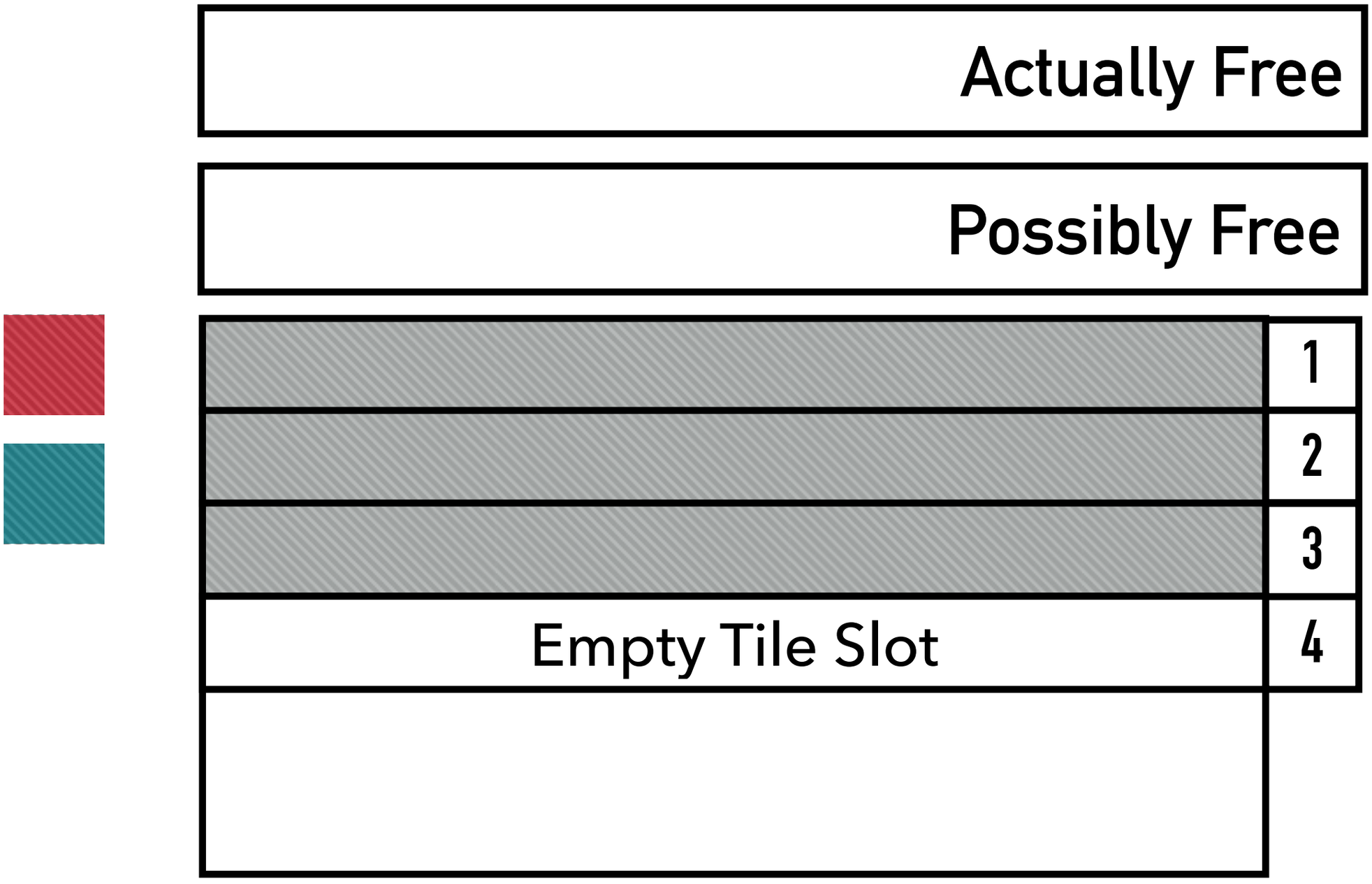
WITHOUT GC



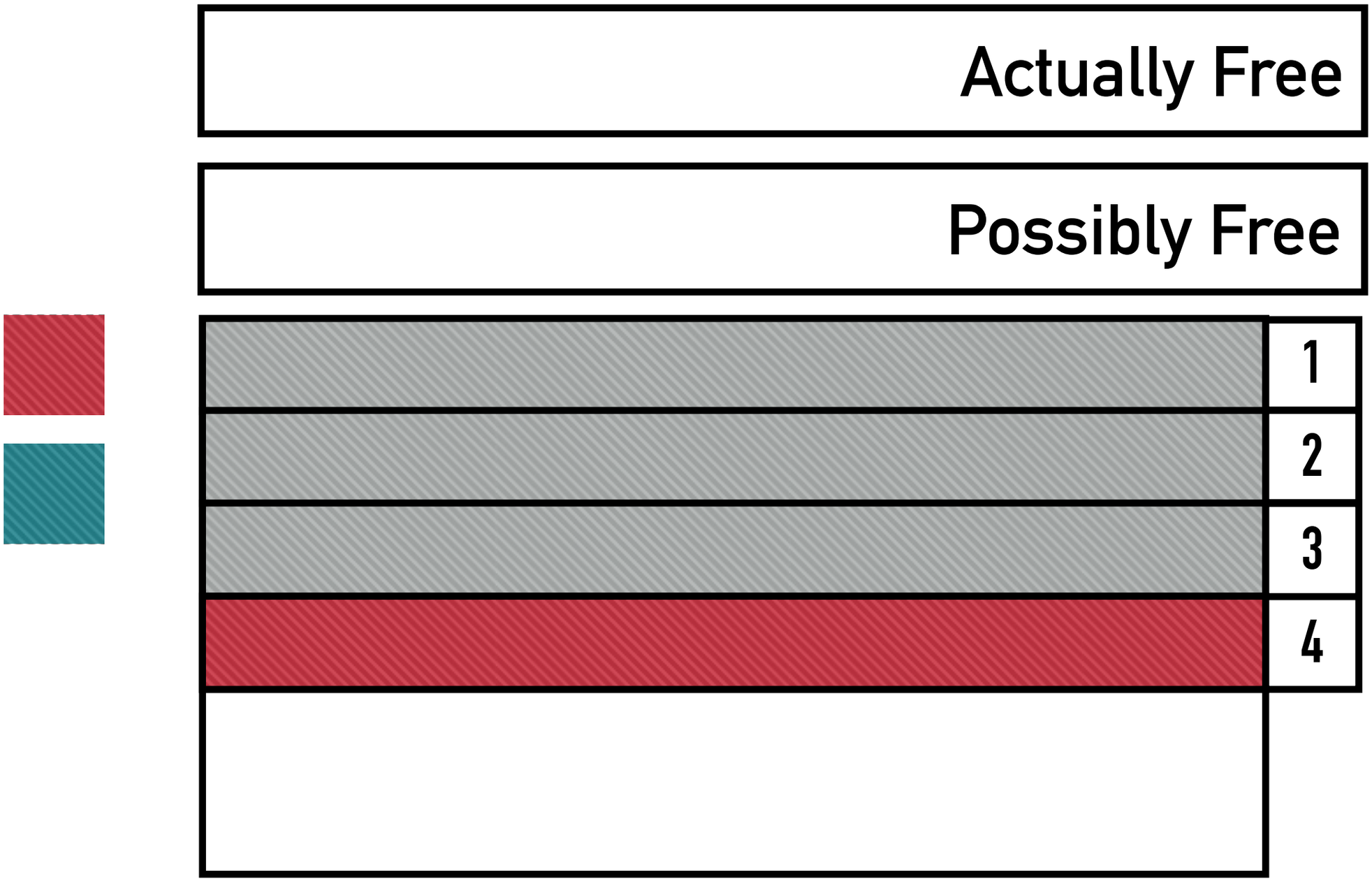
WITHOUT GC



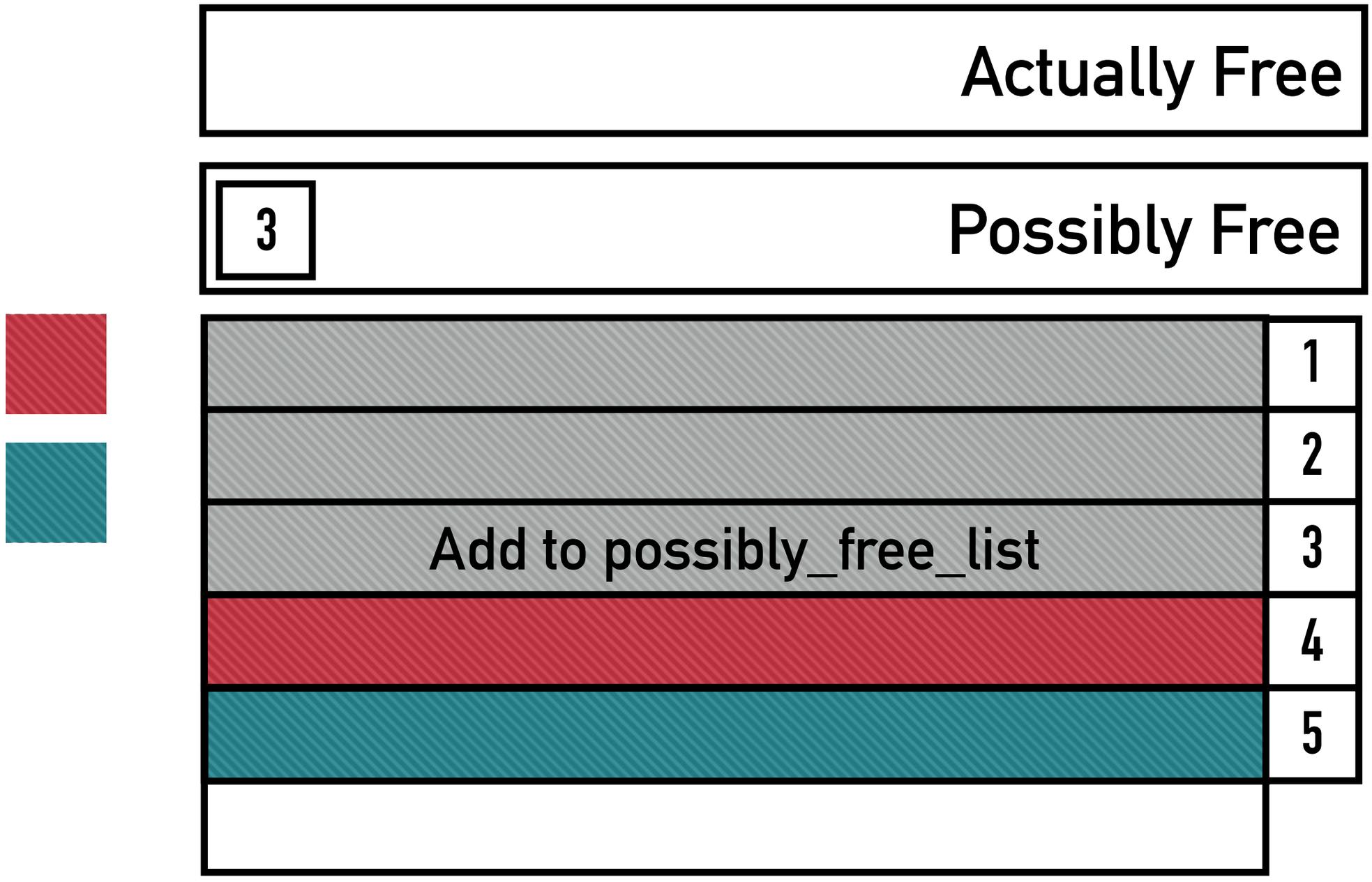
WITH GC


Actually Free
Possibly Free
1
2
3
Empty Tile Slot
4
Tile Group

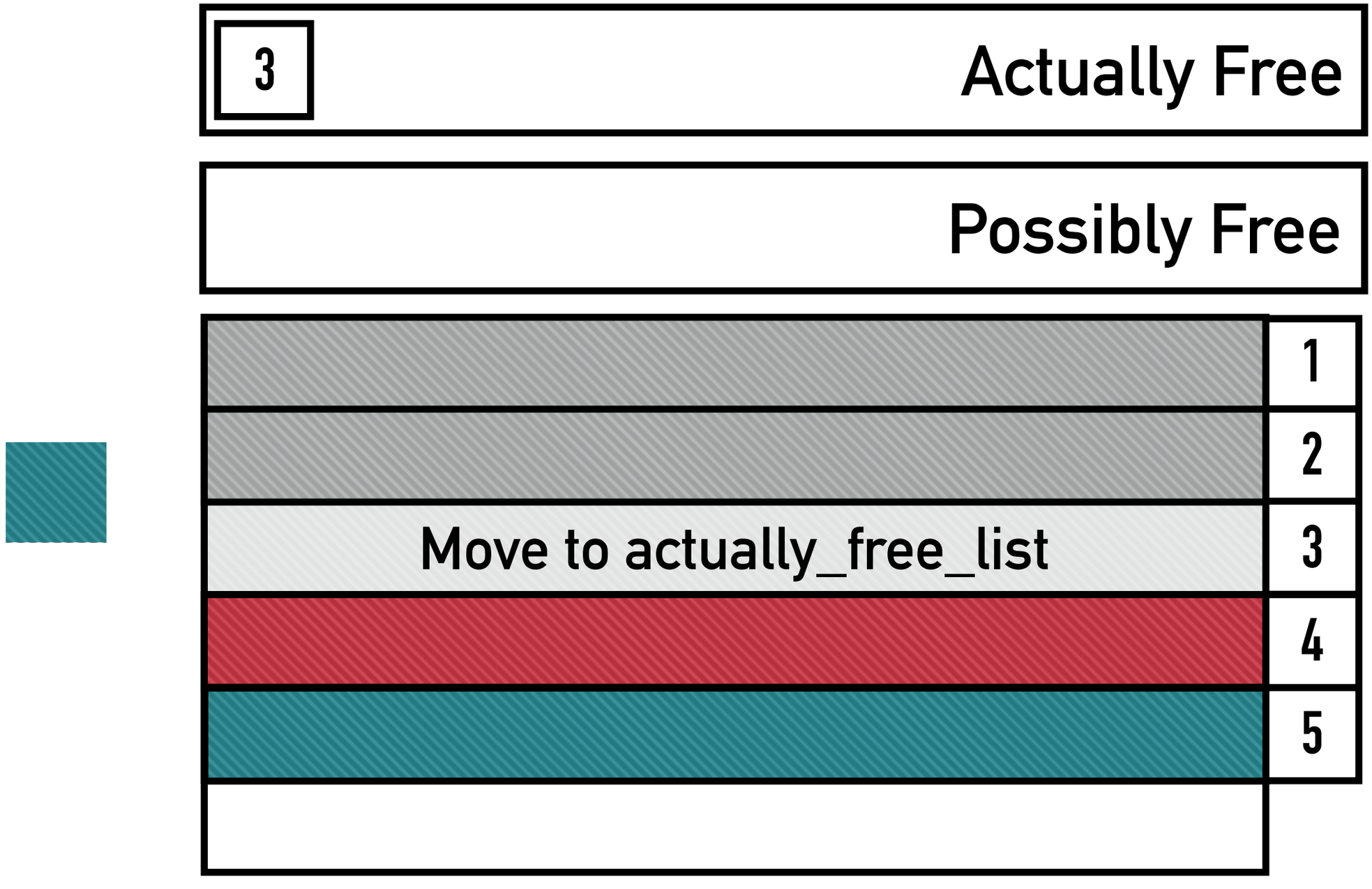
WITH GC



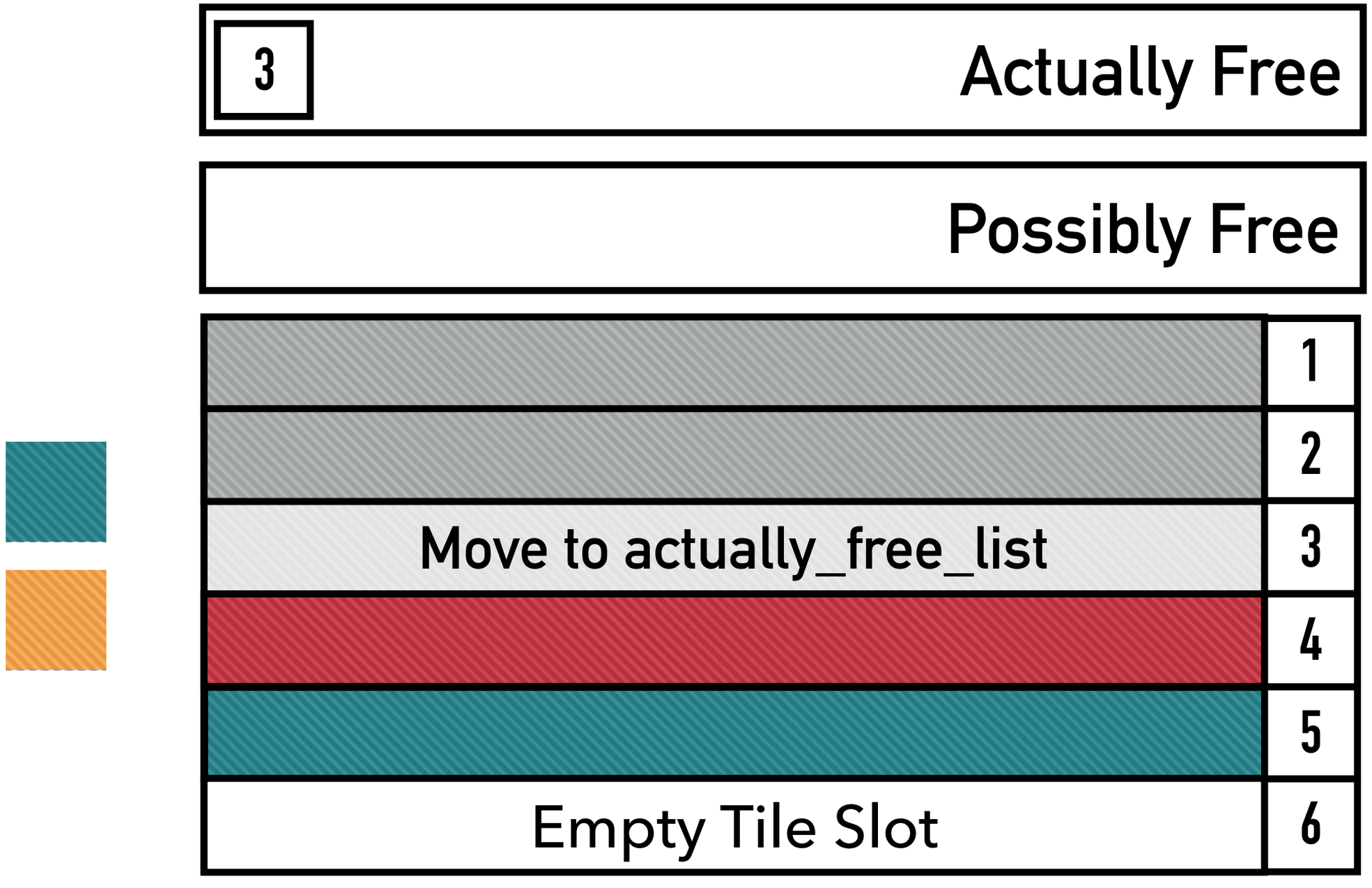
WITH GC


Actually Free
3
Possibly Free
1
2
Add to possibly_free_list
3
4
5
Tile Group

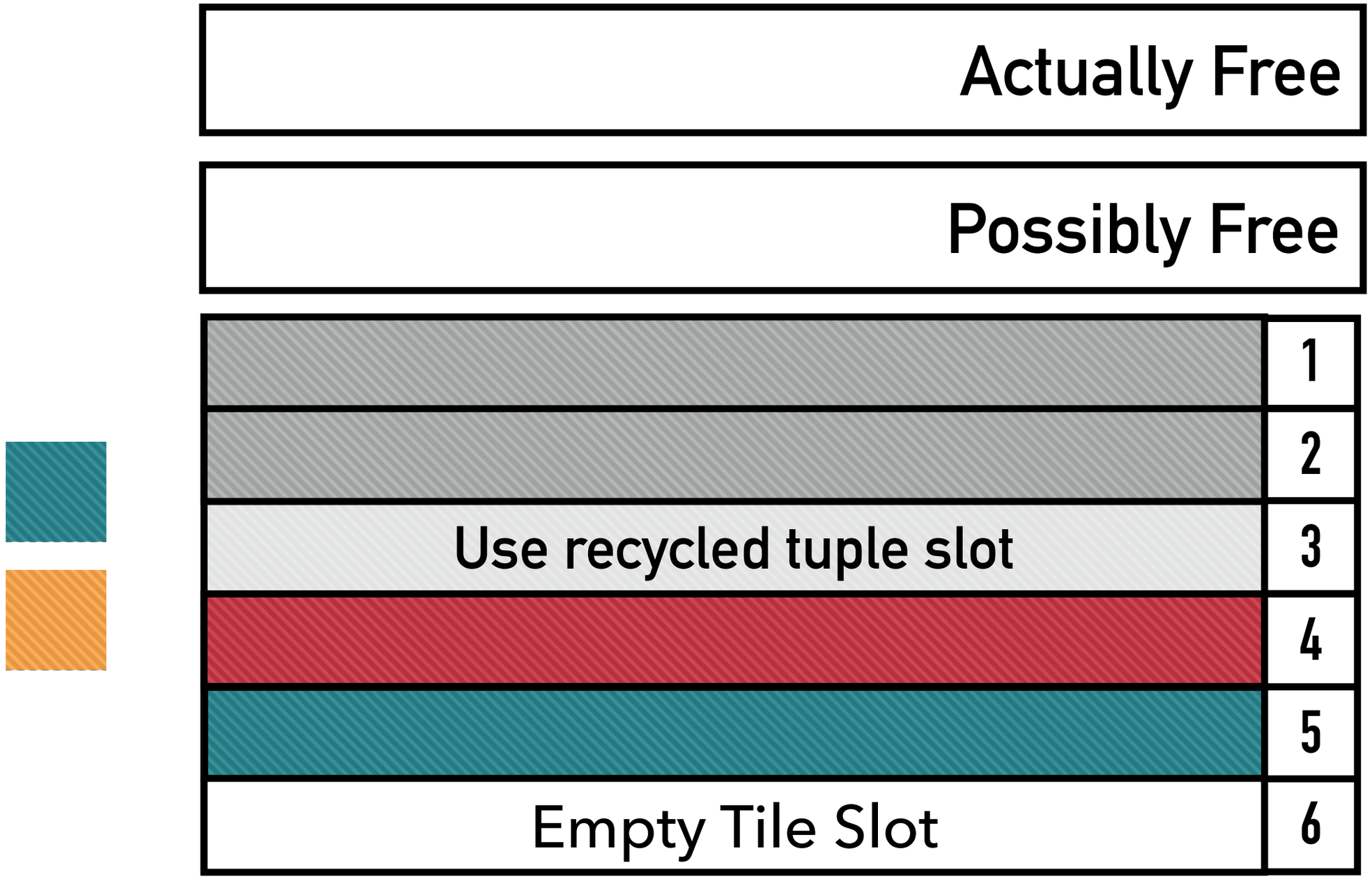
WITH GC


3
Actually Free
Possibly Free
1
2
Move to actually_free_list
3
4
5
Tile Group

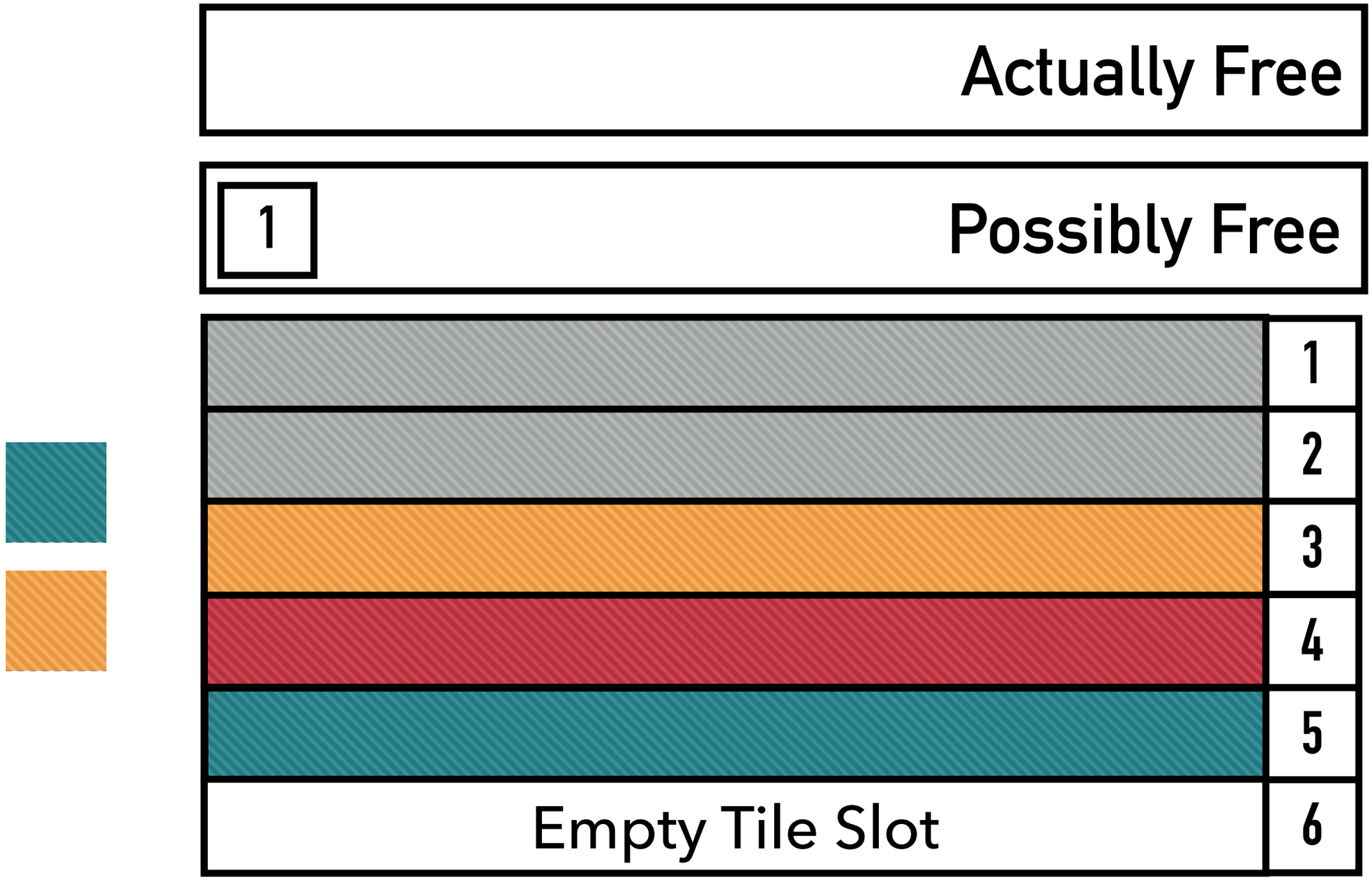
WITH GC


3
Actually Free
Possibly Free
1
2
Move to actually_free_list
3
4
5
Empty Tile Slot
6
Tile Group

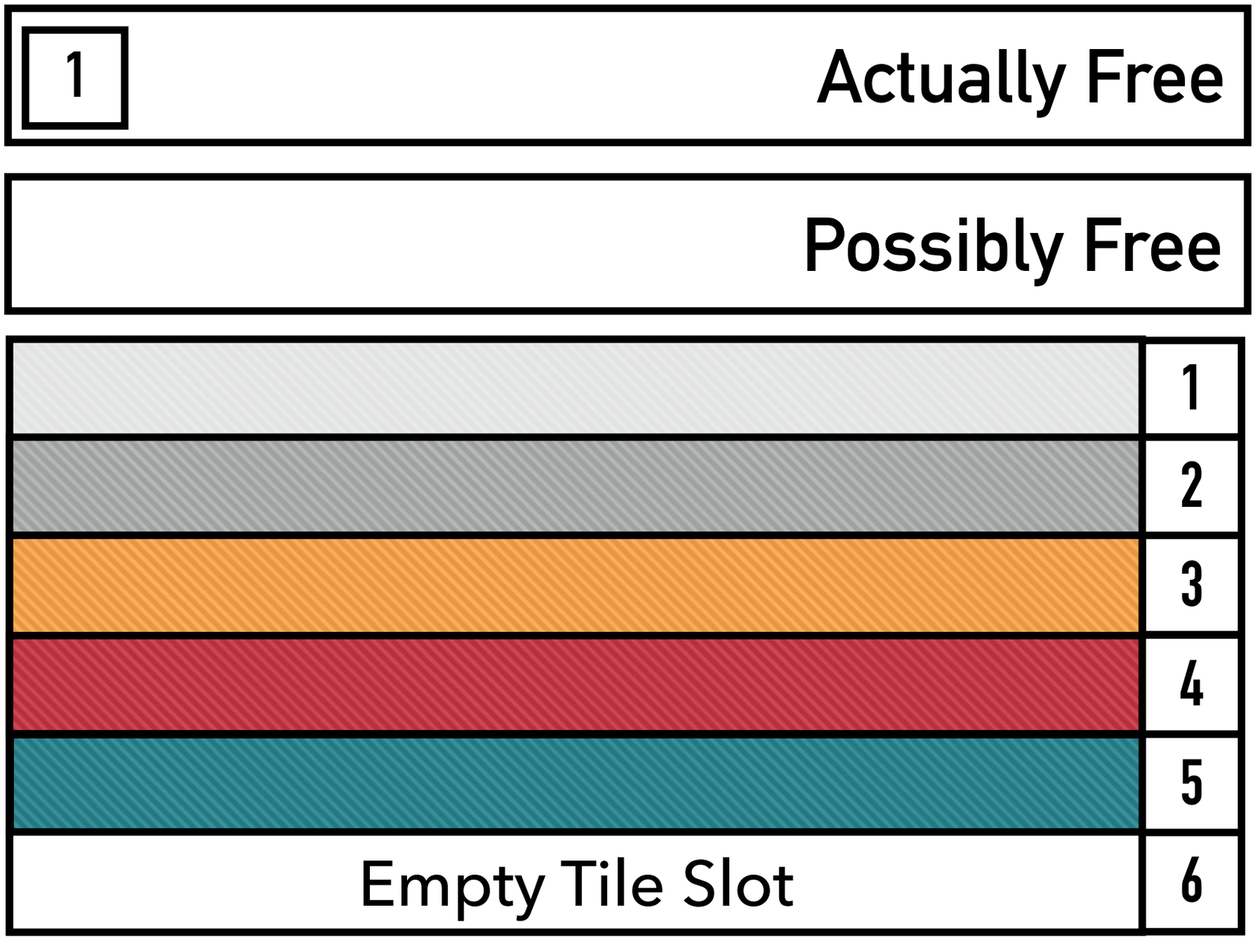
WITH GC


Actually Free
Possibly Free
1
2
Use recycled tuple slot
3
4
5
Empty Tile Slot
6
Tile Group

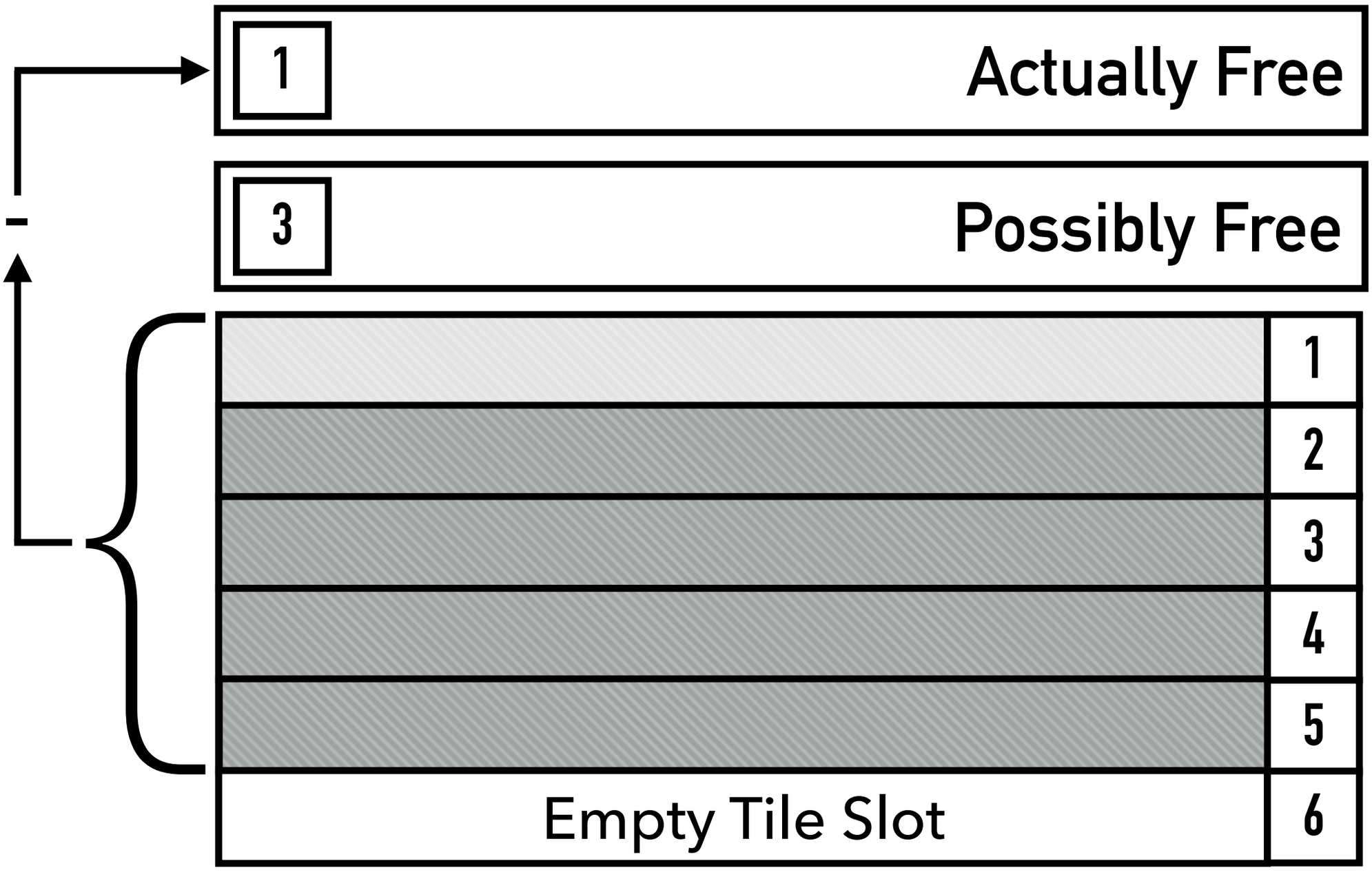
WITH GC


Actually Free
1
Possibly Free
1
2
3
4
5
Empty Tile Slot
6
Tile Group

WITH GC


1
Actually Free
Possibly Free
1
2
3
4
5
Empty Tile Slot
6
Tile Group

MEMORY FOOTPRINT



GC MODES

- Off (default)
- Vacuum
- Naïve co-operative
- Epoch based co-operative

WORKLOAD

- **YCSB** - 80% updates, 20% reads, 10 terminals, 100000 tuples and 100 sec runtime
- **YCSB** - 50% updates, 50% inserts, 10 terminals, 100000 tuples and 100 sec runtime

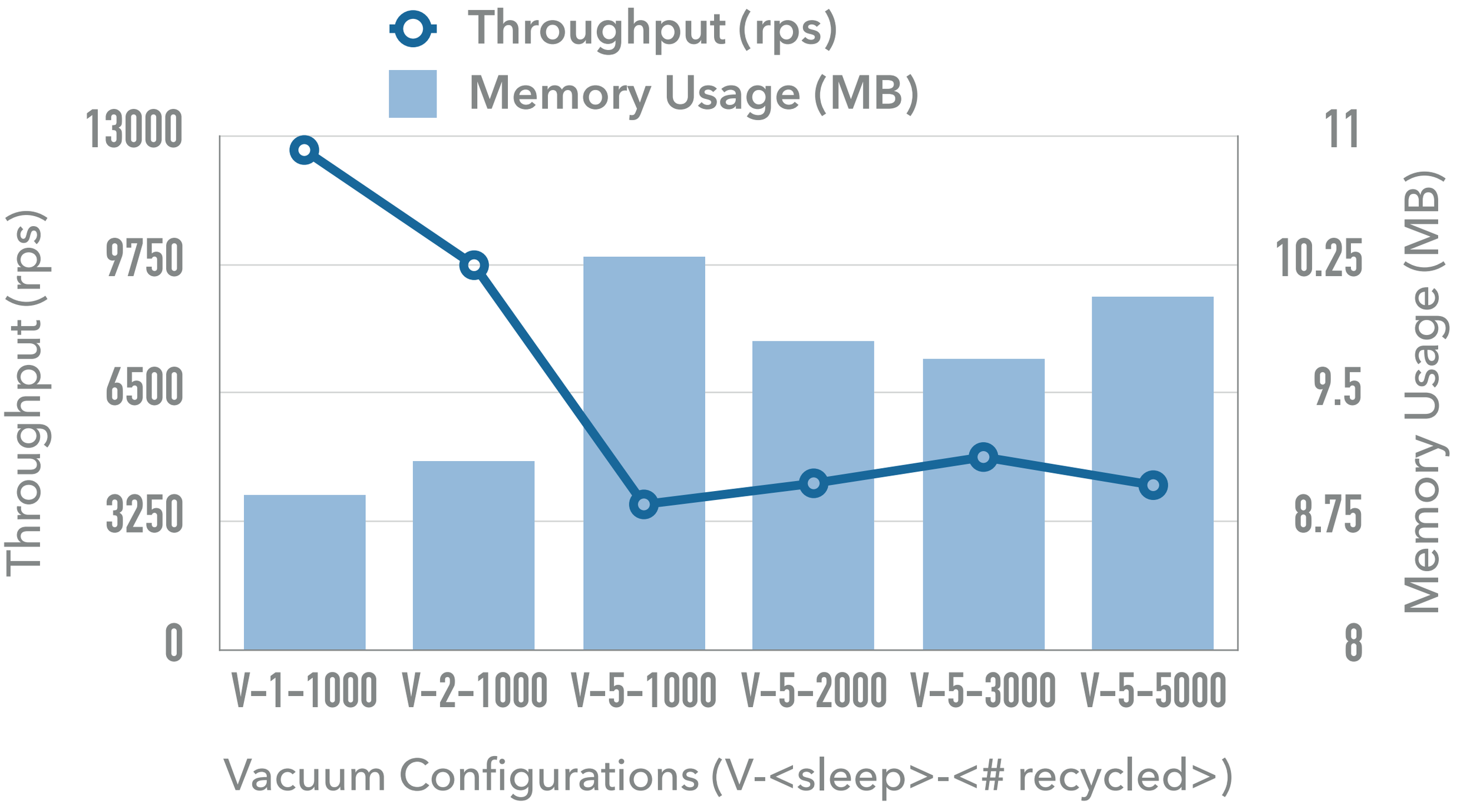
VACUUM

- Separate thread started as Peloton bootstraps
- Periodically moves elements from possibly_free_list to actually_free_list

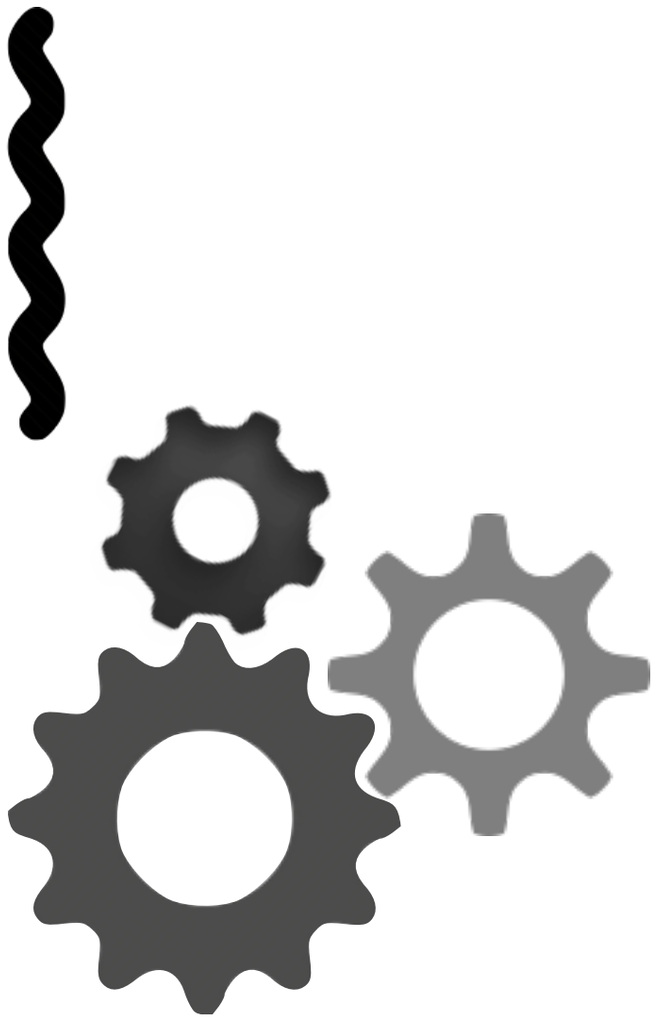
VACUUM TRADEOFFS

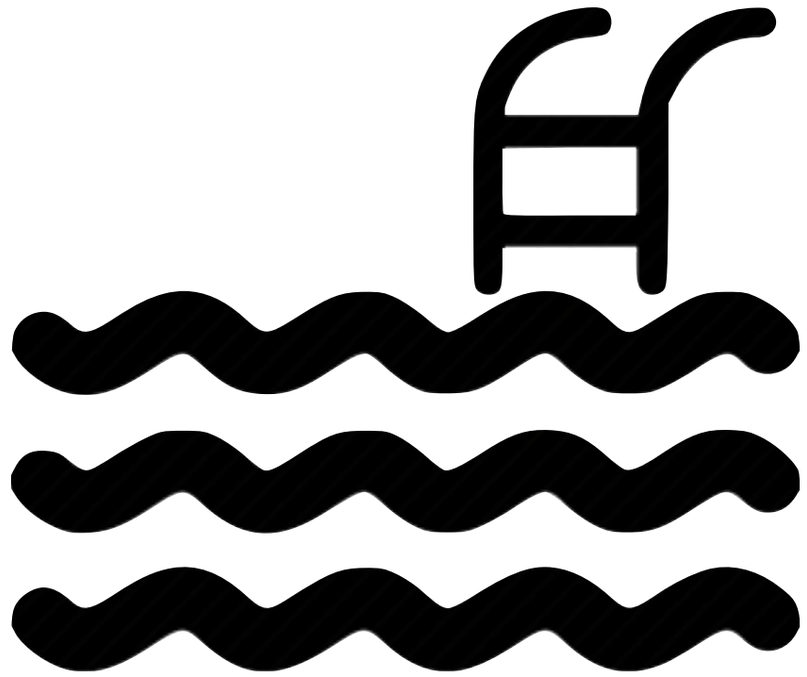
- Vacuum thread period vs GC efficiency
- # Recycled per vacuum invocation vs GC efficiency

VACUUM TRADEOFFS MEASURED

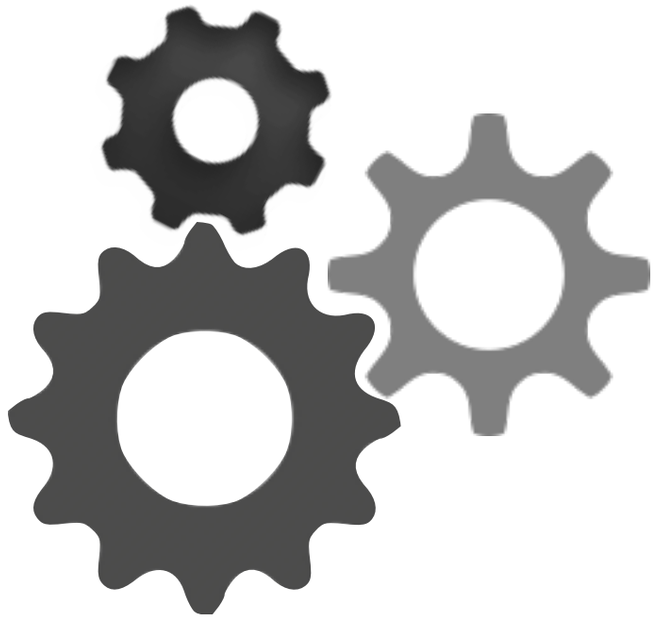


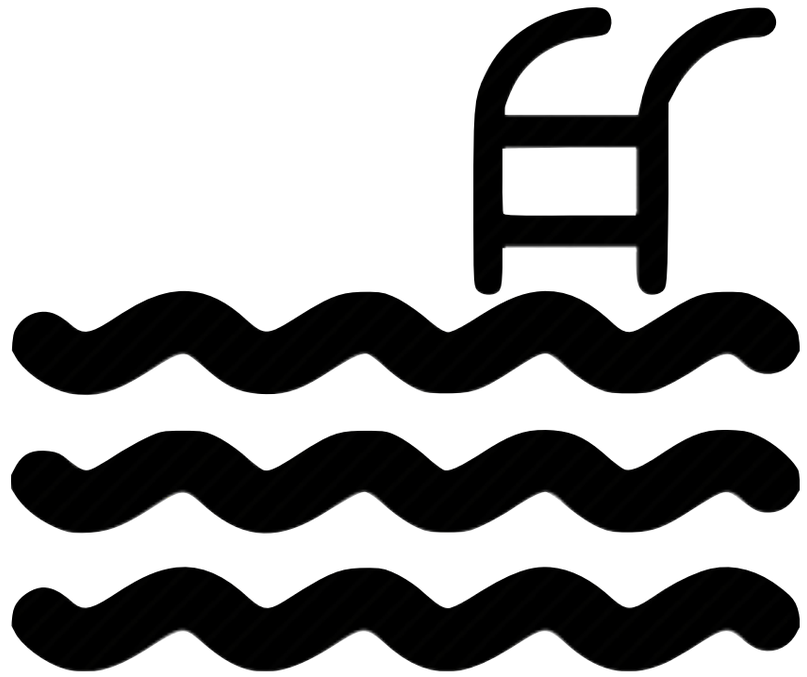
NAÏVE CO-OPERATIVE (COOPERATIVE)





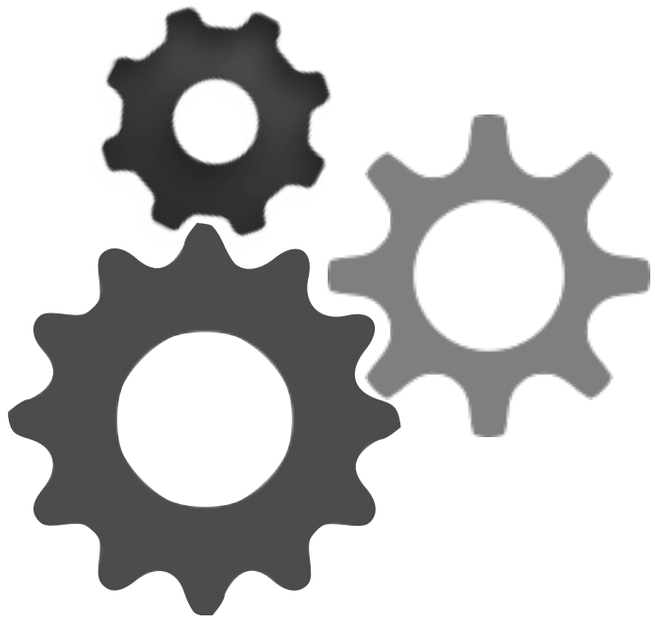
NAÏVE CO-OPERATIVE (COOPERATIVE)

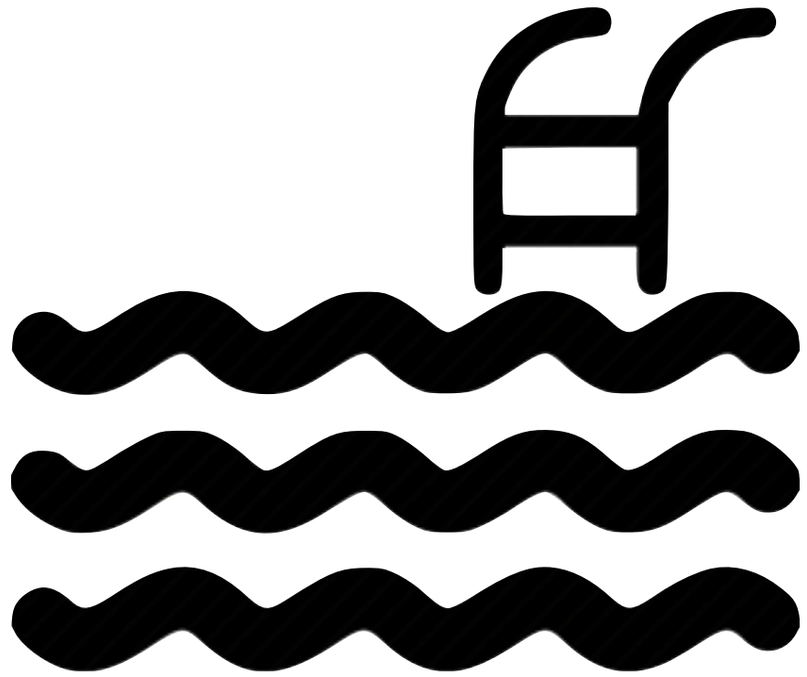




PERFORM GC

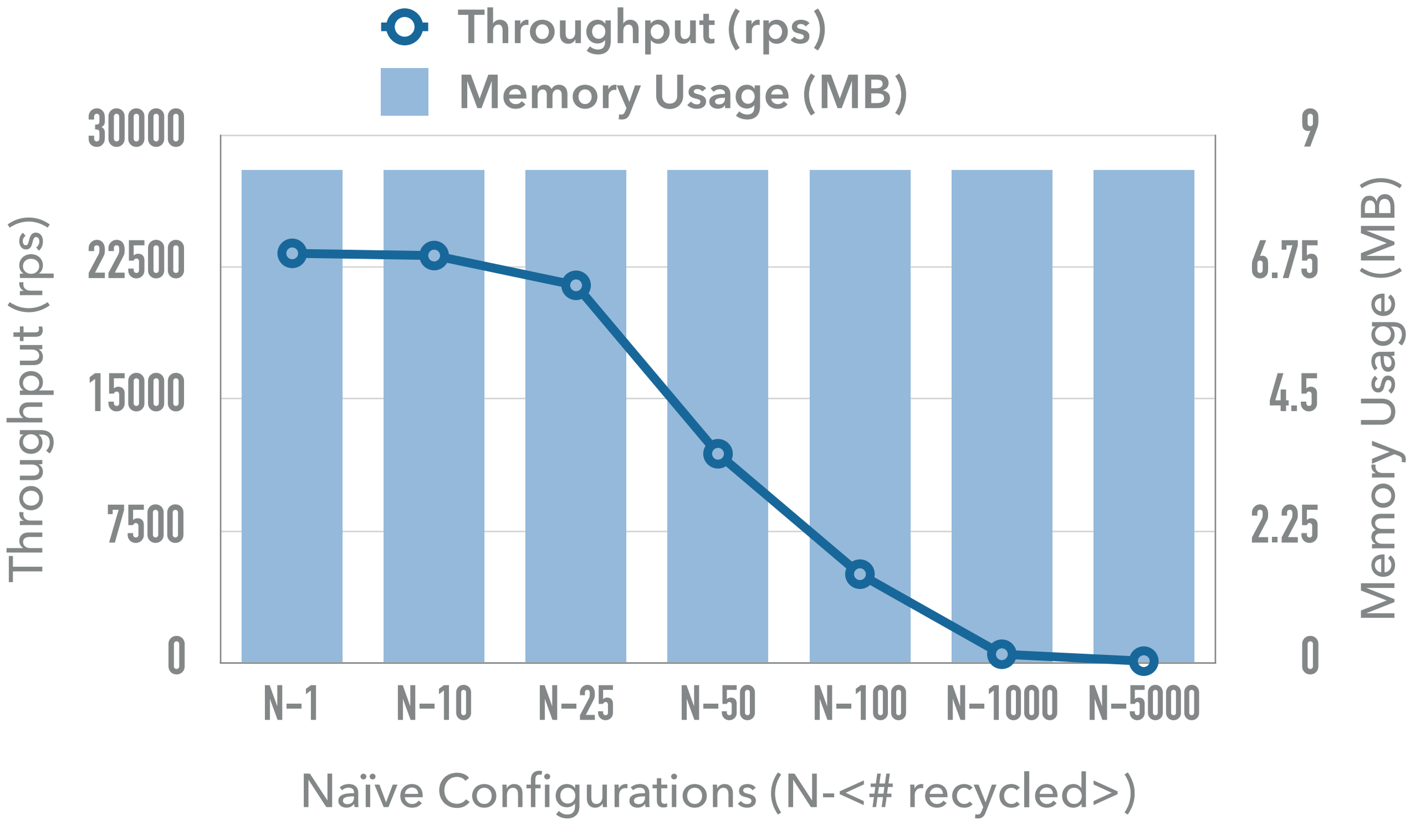
NAÏVE CO-OPERATIVE (COOPERATIVE)



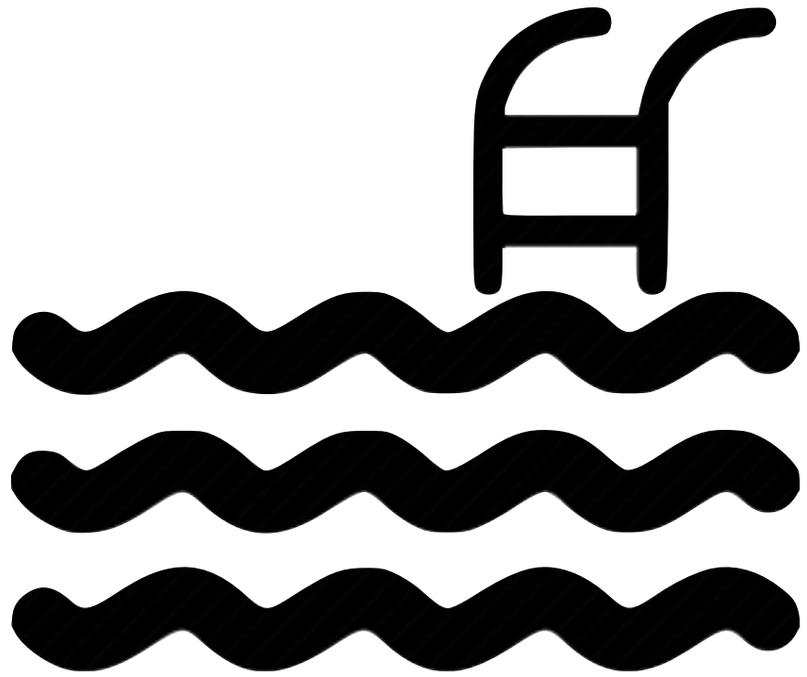


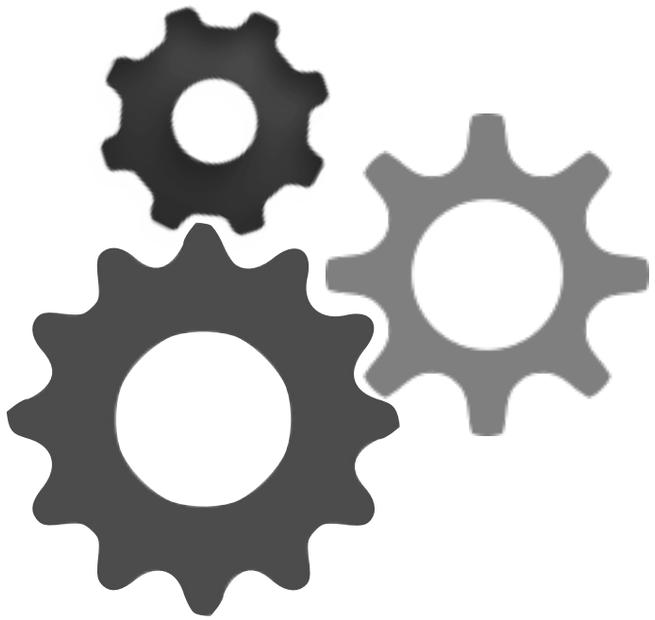
PERFORM GC

COOPERATIVE TRADEOFFS MEASURED

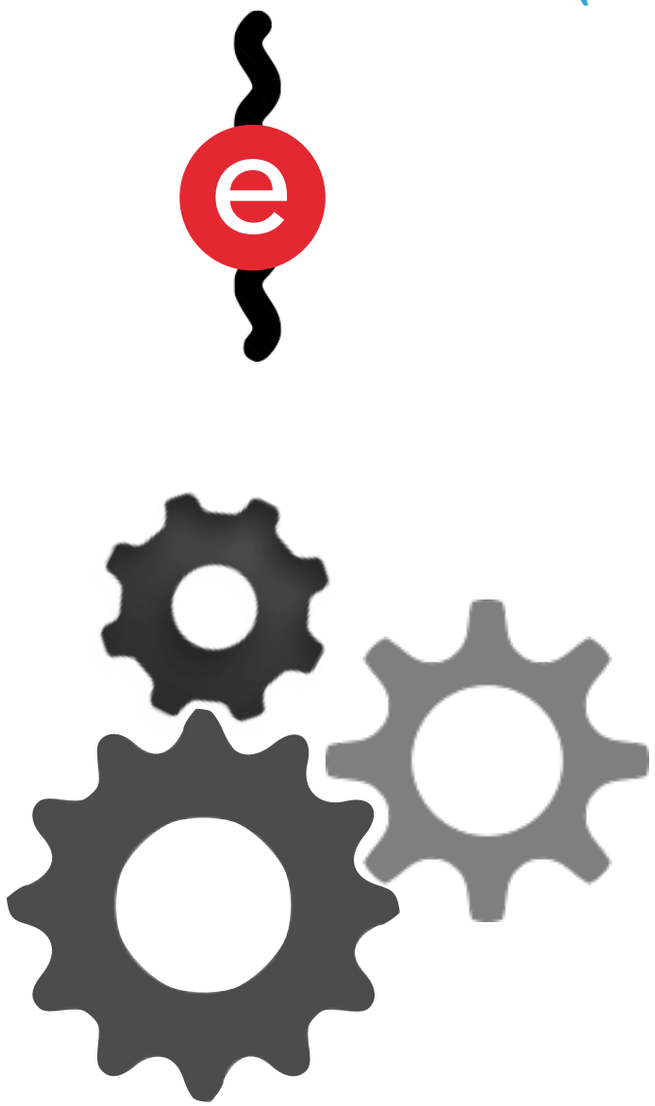


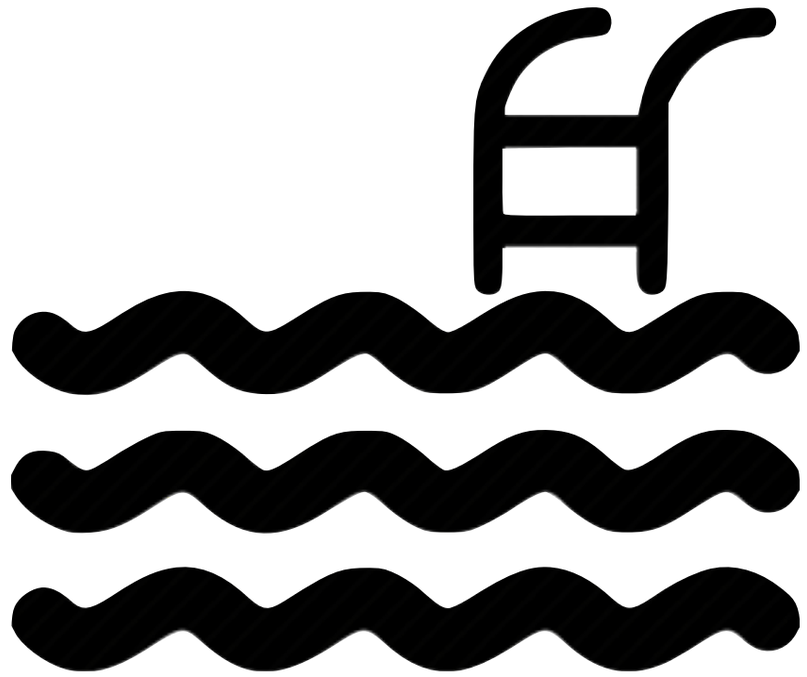
EPOCH CO-OPERATIVE (EPOCH)



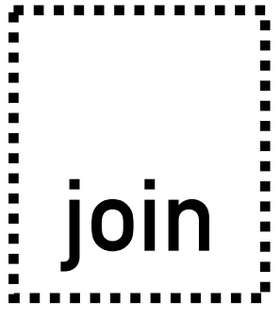


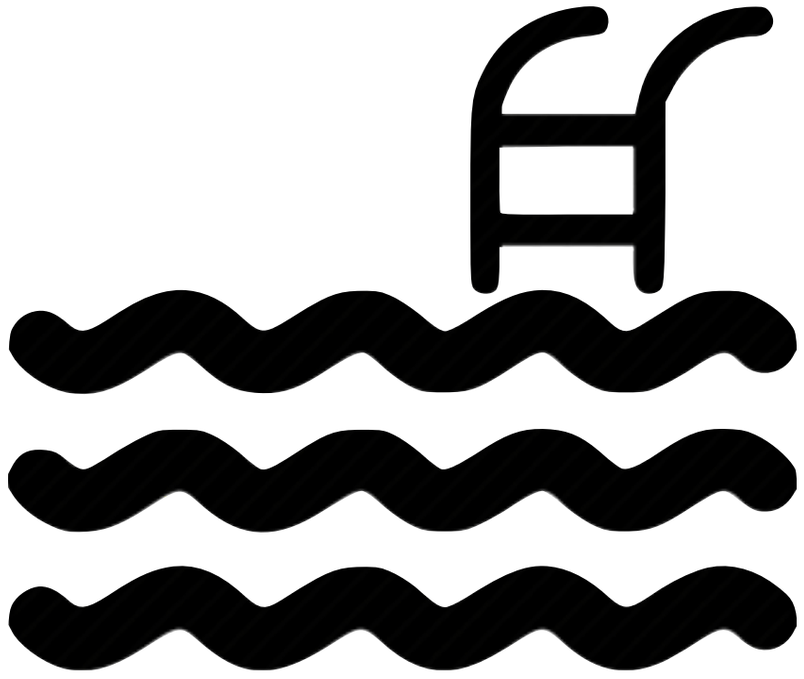
EPOCH CO-OPERATIVE (EPOCH)





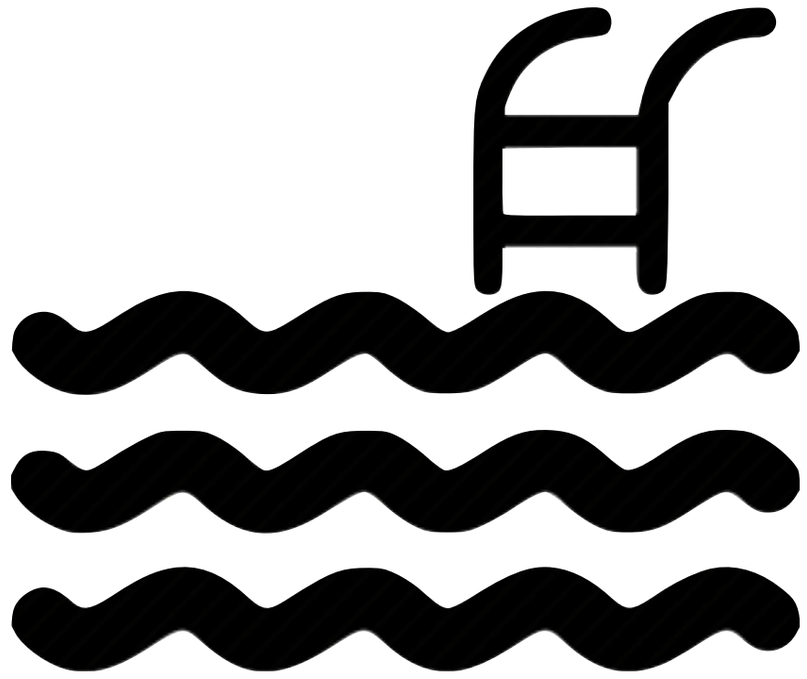
EPOCH CO-OPERATIVE (EPOCH)



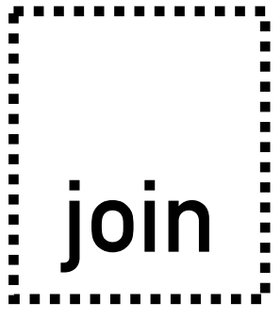


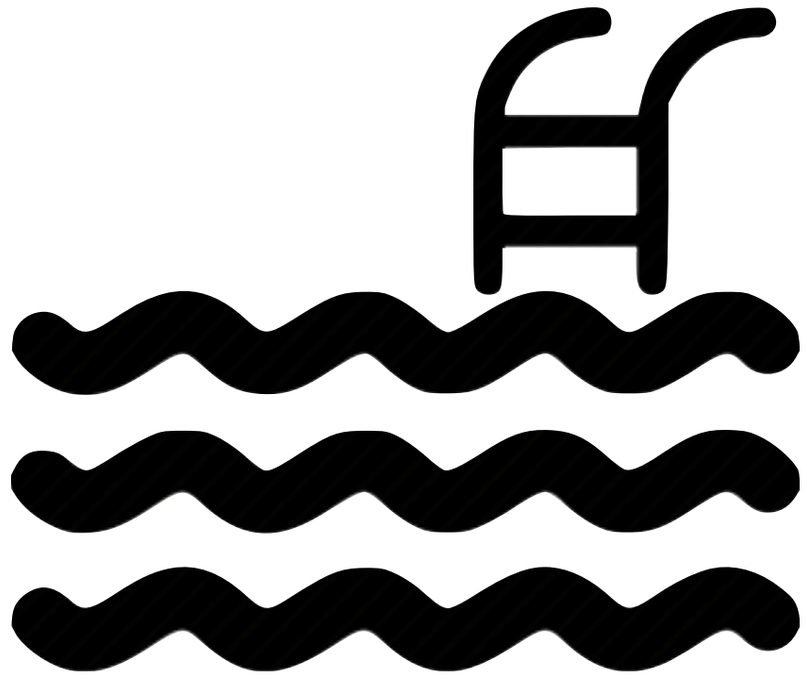
EPOCH CO-OPERATIVE (EPOCH)

join

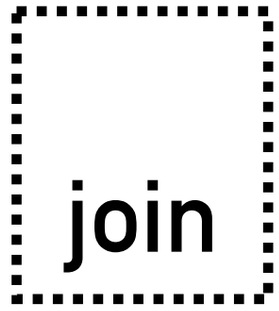


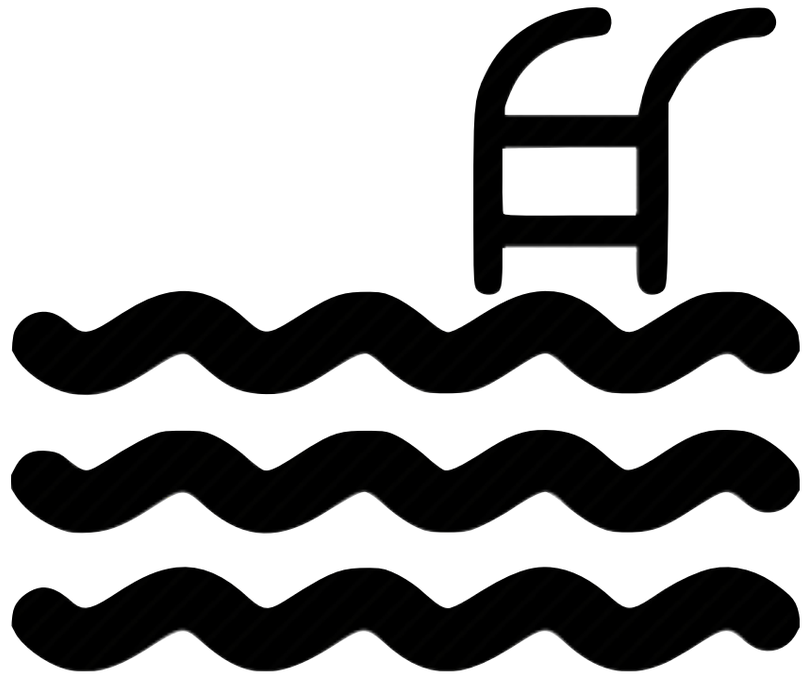
EPOCH CO-OPERATIVE (EPOCH)



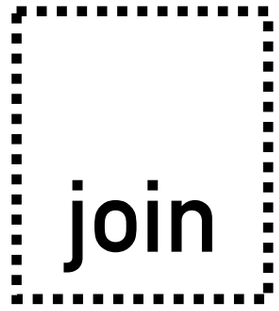


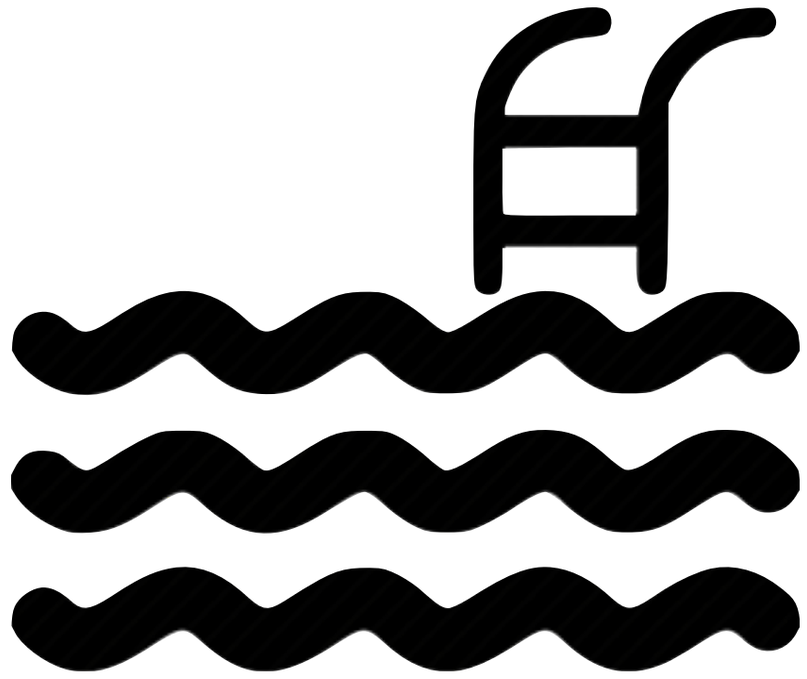
EPOCH CO-OPERATIVE (EPOCH)

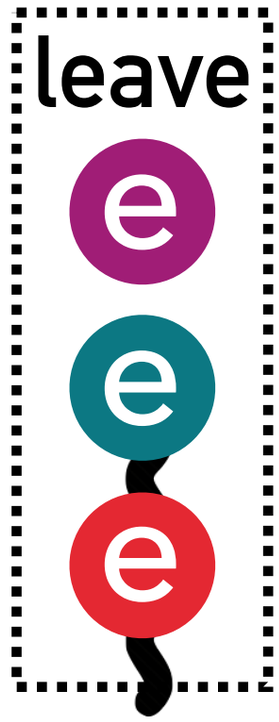




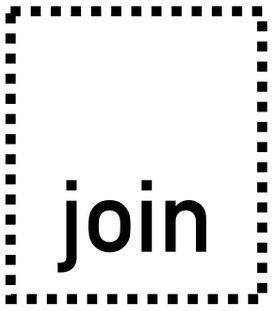
EPOCH CO-OPERATIVE (EPOCH)

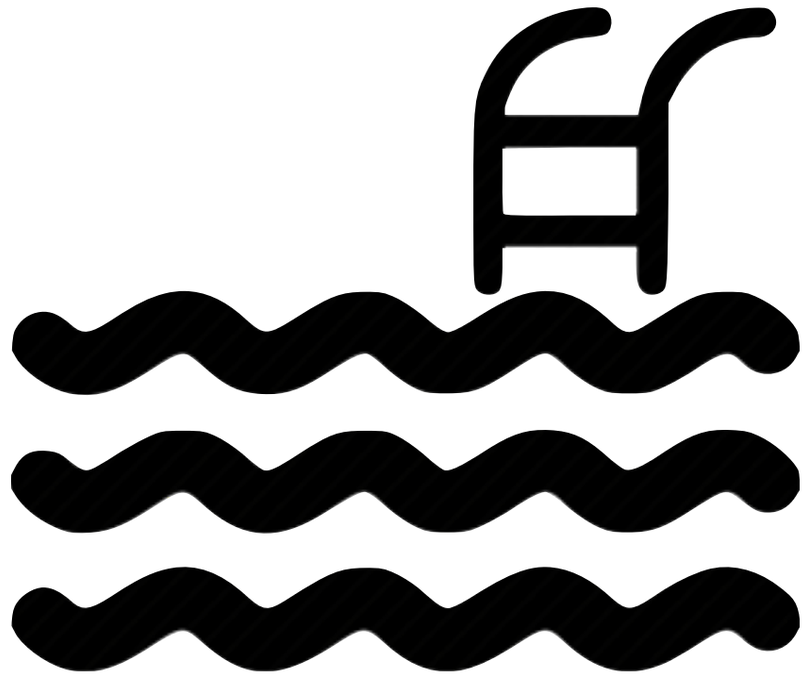


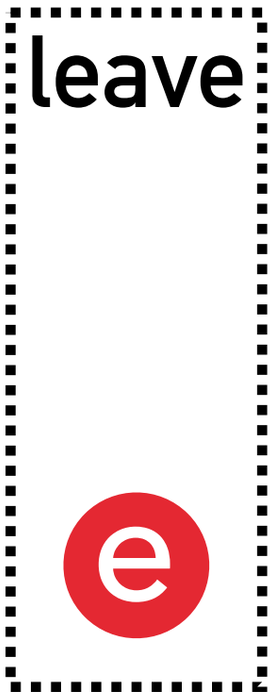


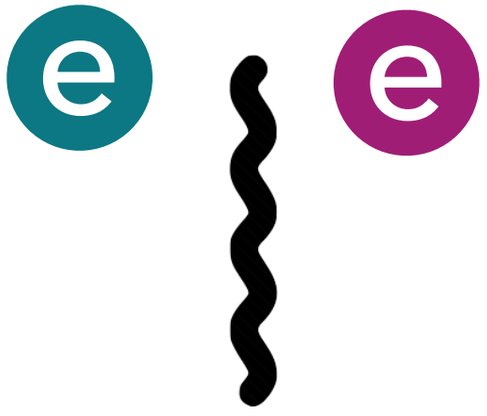


EPOCH CO-OPERATIVE (EPOCH)

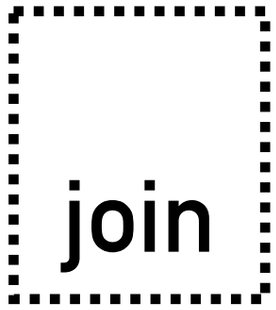


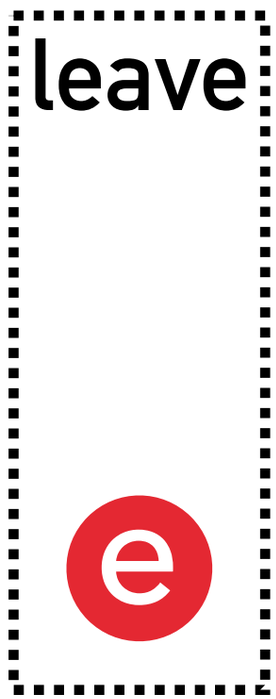


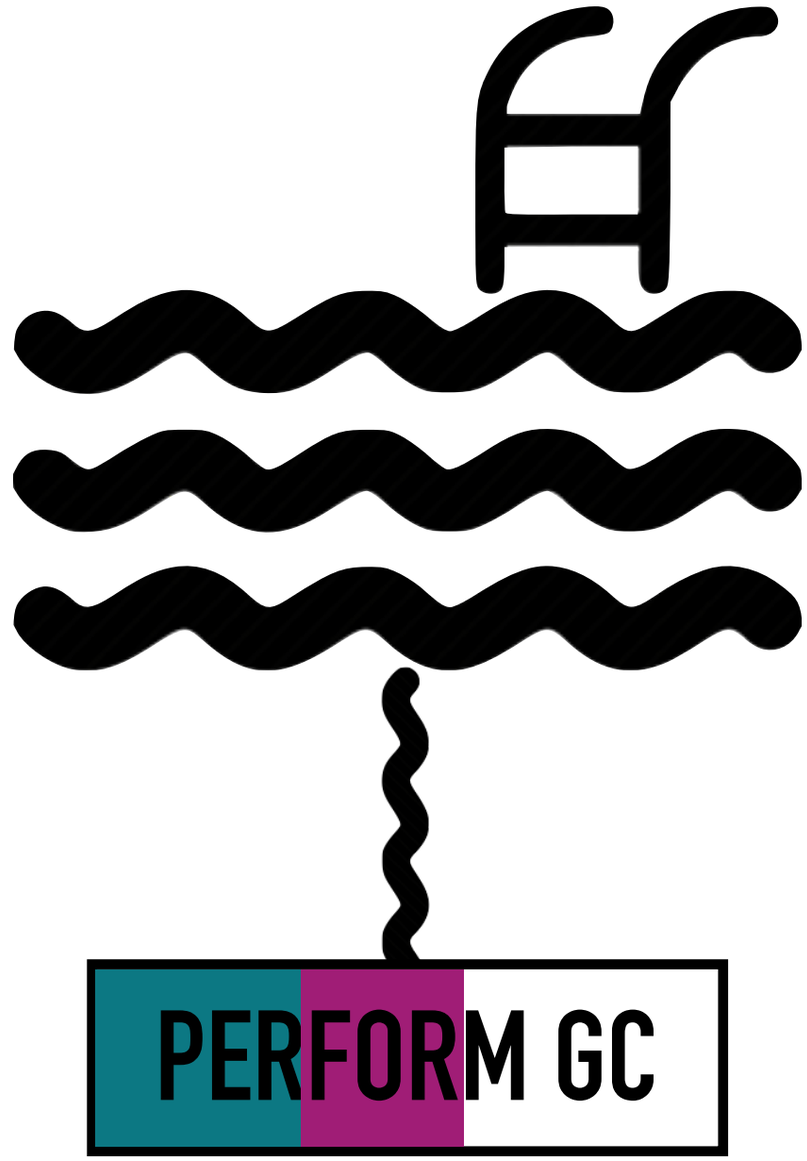




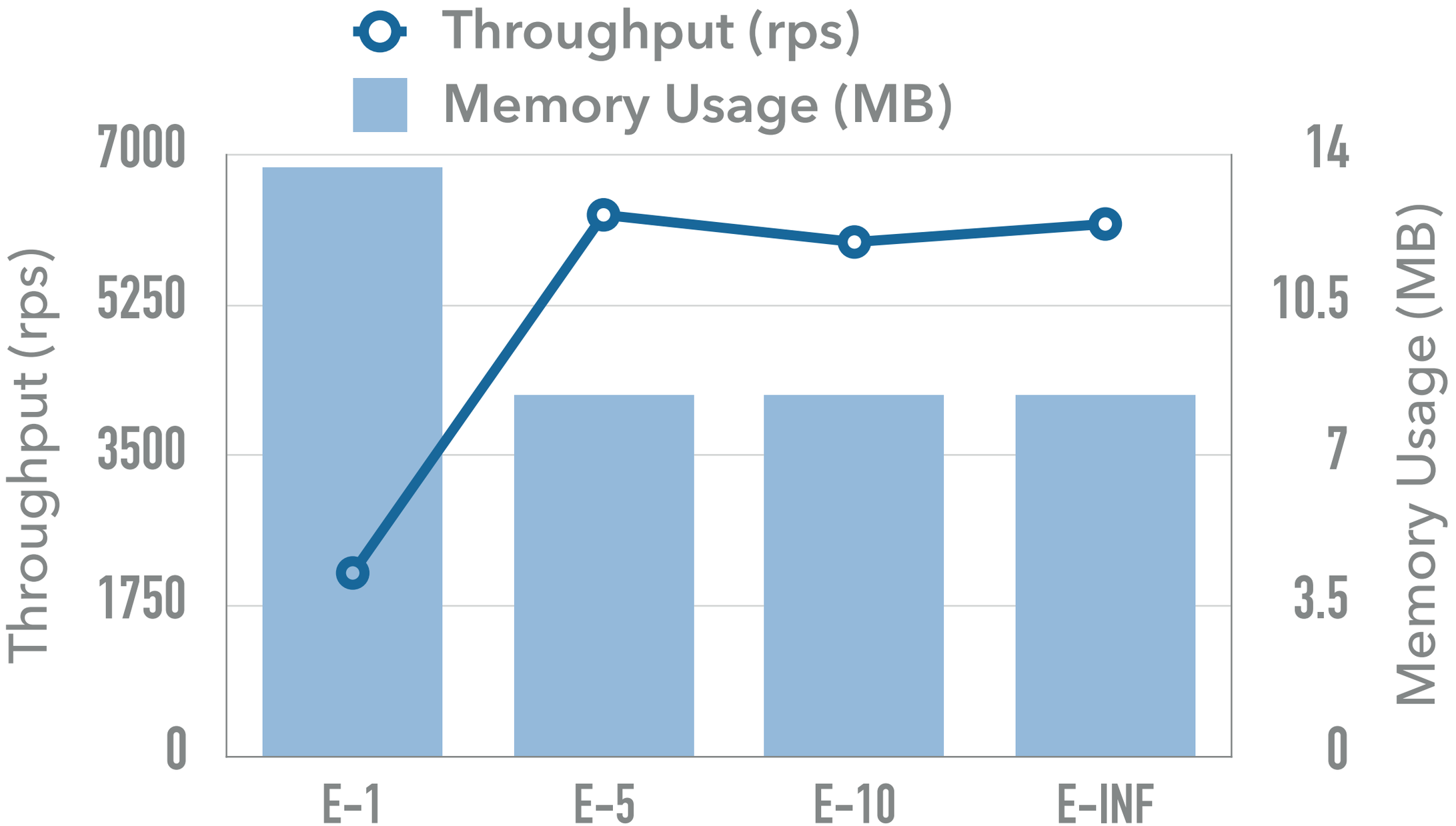
EPOCH CO-OPERATIVE (EPOCH)



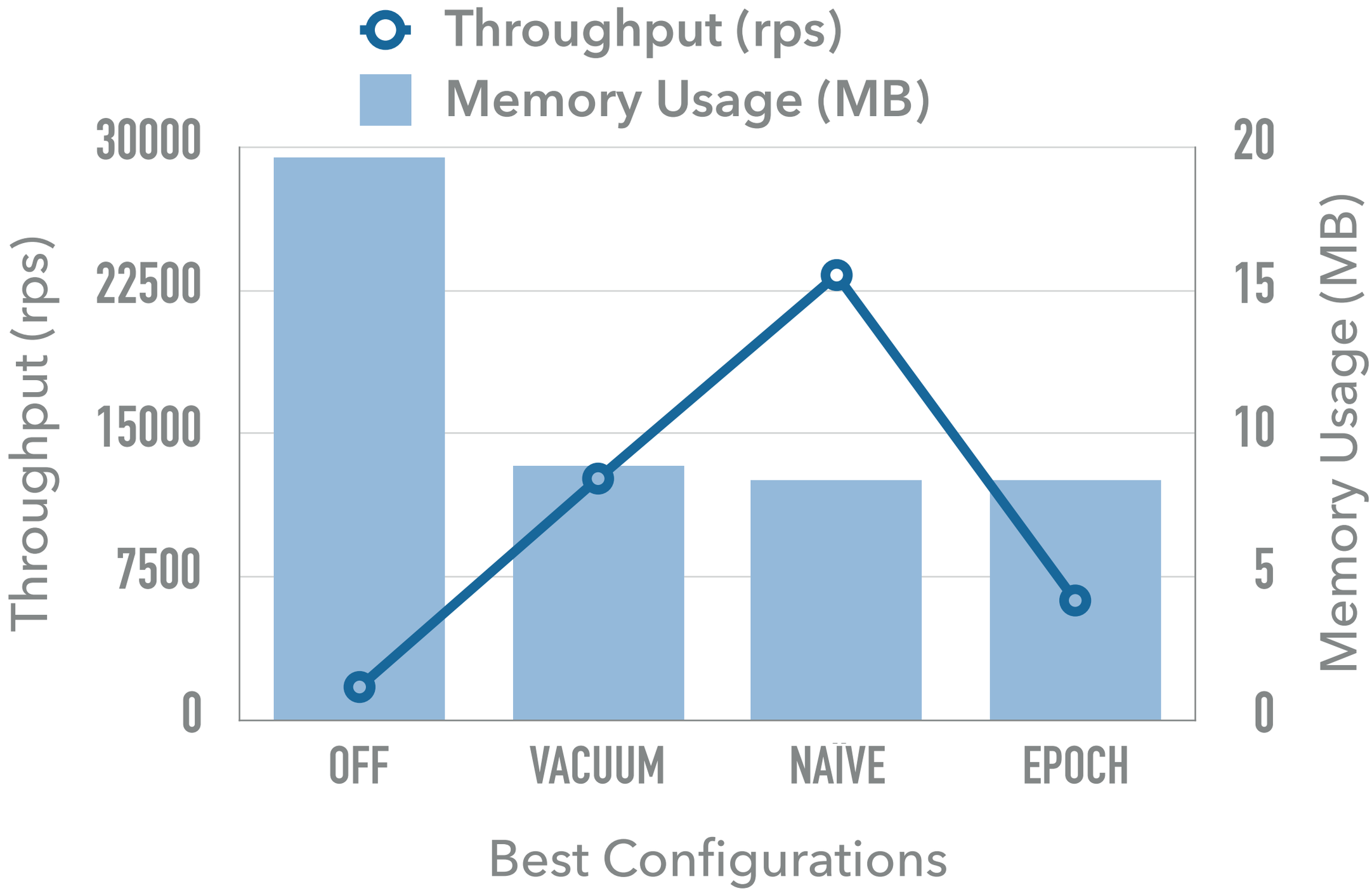




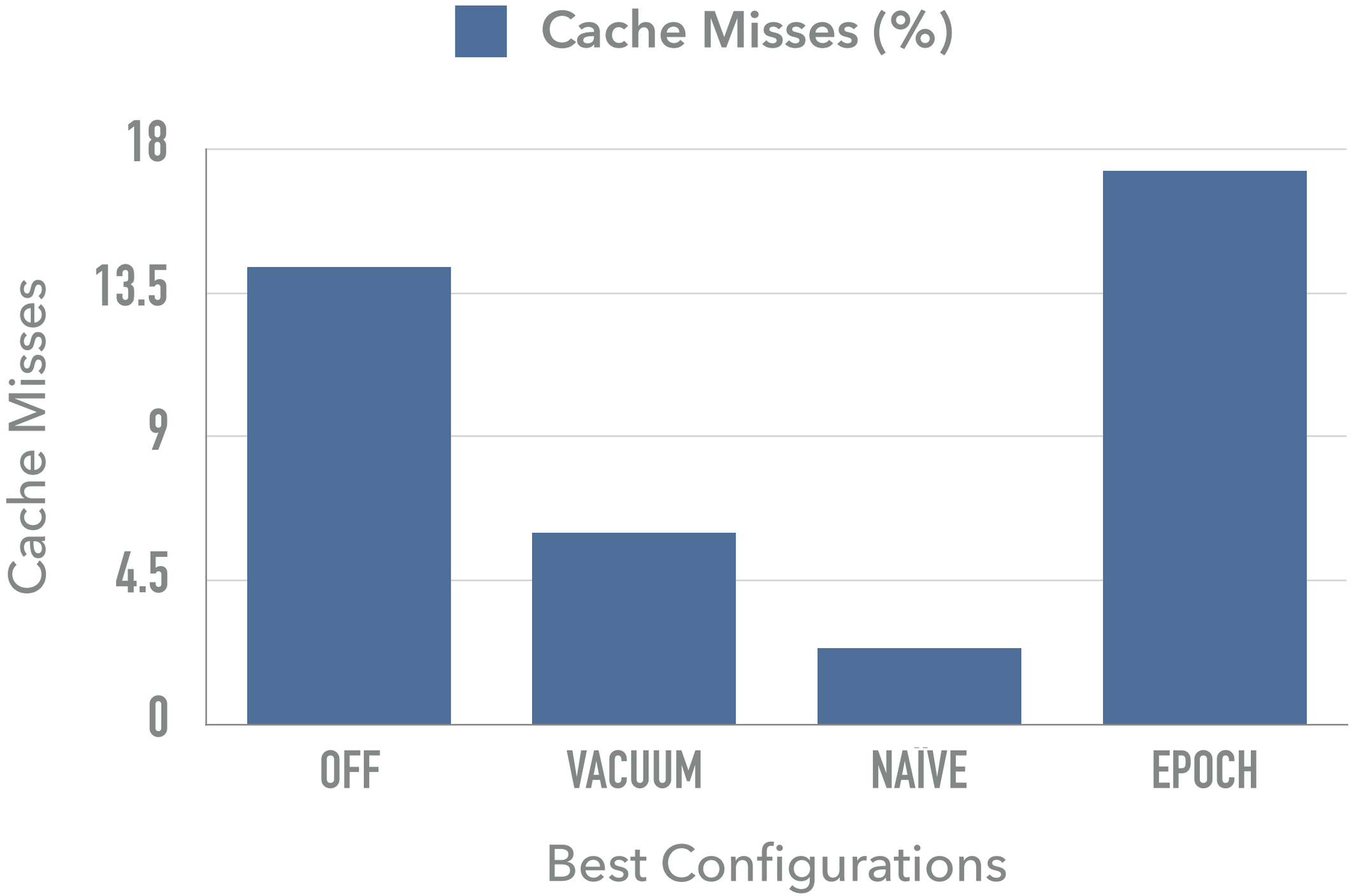
EPOCH TRADEOFFS MEASURED



OVERALL GC COMPARISON



CACHE MISSES MATTER AND THEY DON'T!



ENTIRE TABLE TRUNCATIONS

- Possibly_free_list may potentially end up with an absurdly large number of free slots
- Has to be handled as a special case

RECYCLING ACROSS TILE GROUPS

- Current recycling is at table granularity - i.e. across tile groups
- Different tile group schemas in the same table may become problematic
- Tradeoff: recycling granularity vs # tuples recycled

QUESTIONS?

Thank You. . .