Hot Restart Persistence

MARKO TOPOLNIK

Hazelcast IMDG as a distributed cache

- cluster that holds cached data in RAM
  - result of any kind of expensive computation
- use cases:
  - rendered HTML snippets
  - DB query results
  - results of transactions distributed over several back-end systems

Reasons to shut down and restart

- cluster-wide power outage
- orderly shutdown for maintenance
  - replace some classes on the classpath
  - change a backend password
  - annual poweroff
  - test resilience to outages

Problems with restarting

- data per node reaching 1 TB
- caches are cold after restart
- could take hours or even days to re-heat
- larger concern than downtime itself

Solution: Hot Restart

- persistence layer to back up RAM data
- pure writing, then pure sequential reading
- no random lookup by key
- must support bulk delete
- low latency, high throughput is a must
- ⇒ different design than general-purpose DB

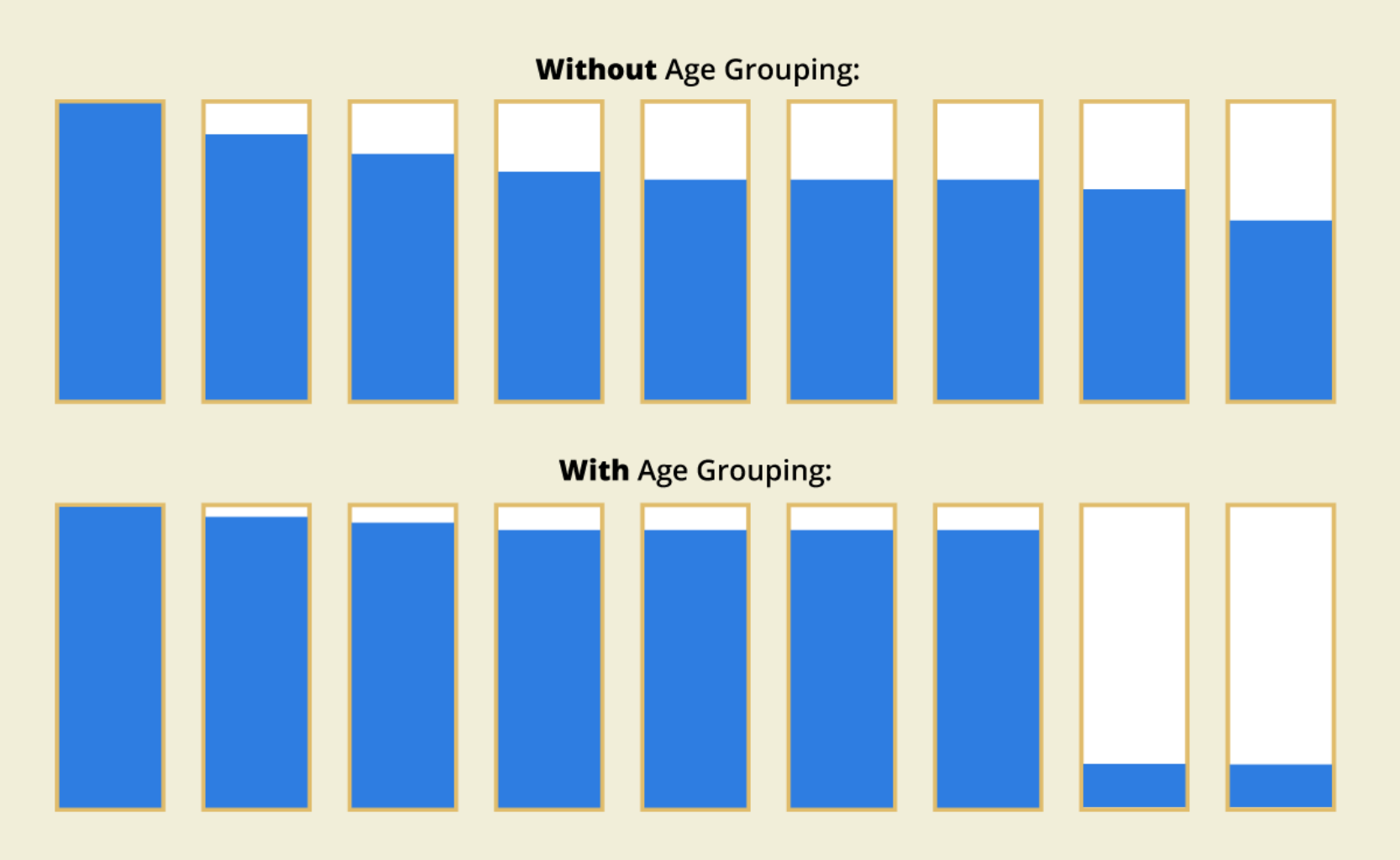
Design details: updating

- log-structured storage
- inspired by the *Sprite LFS* paper
- http://web.stanford.edu/~ouster/cgi-bin/papers/lfs.pdf
- a log of all update events
  - *put* → value record
  - *remove* → tombstone
  - *clear by prefix* → prefix tombstone
- after restart: replay all updates to the RAM store
- needs GC (write amplification)

Design details: GC

- **concurrent**: in the background, responds to change
- **incremental**: only a few chunks GC'd at a time
- **generational**:
  - goal: collect much garbage, copy little live data
  - data grouped by age
  - outcome: bimodal garbage distribution
- **no file reading**: RAM-to-file
  - except tombstone GC

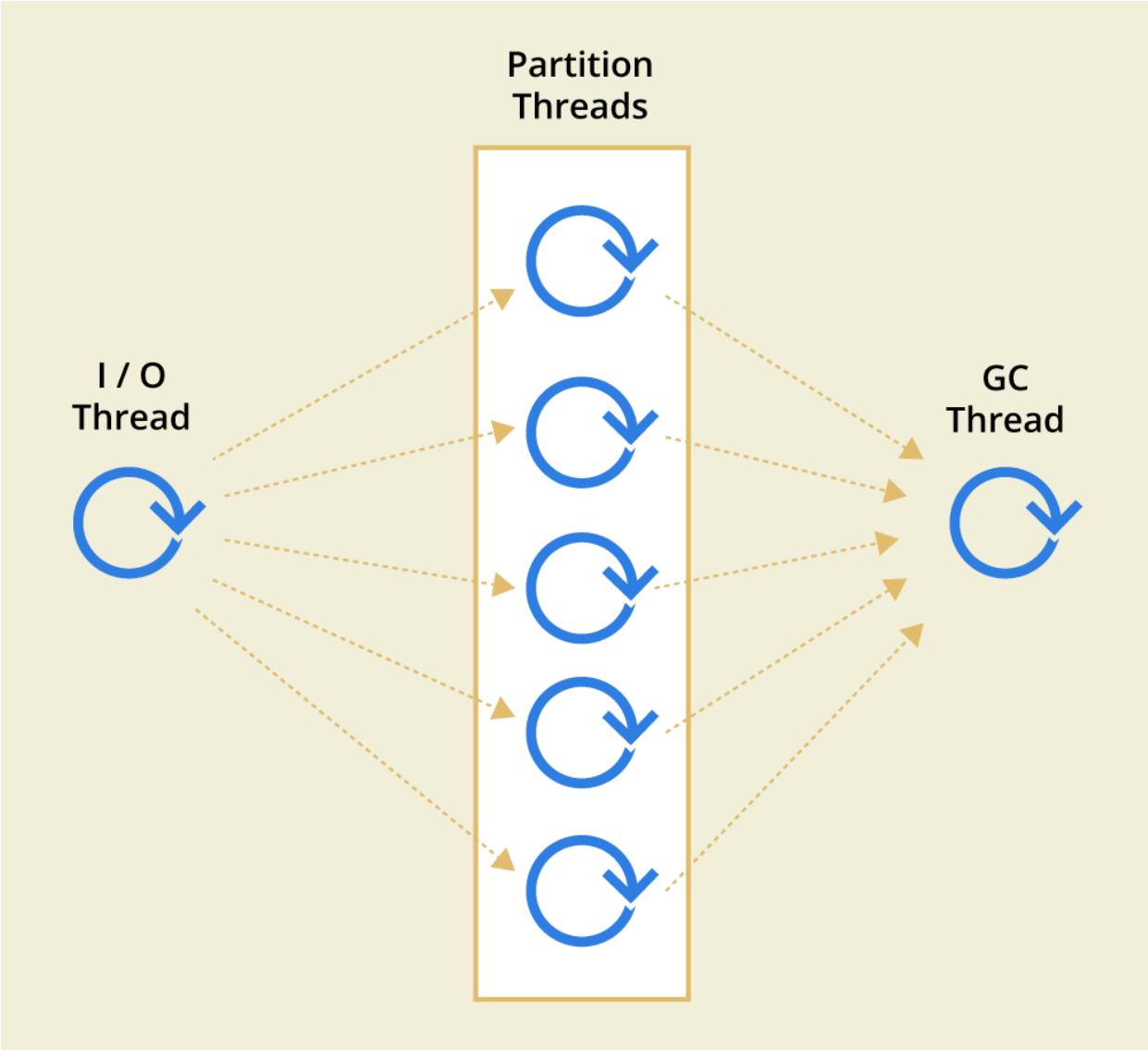
Grouping by age



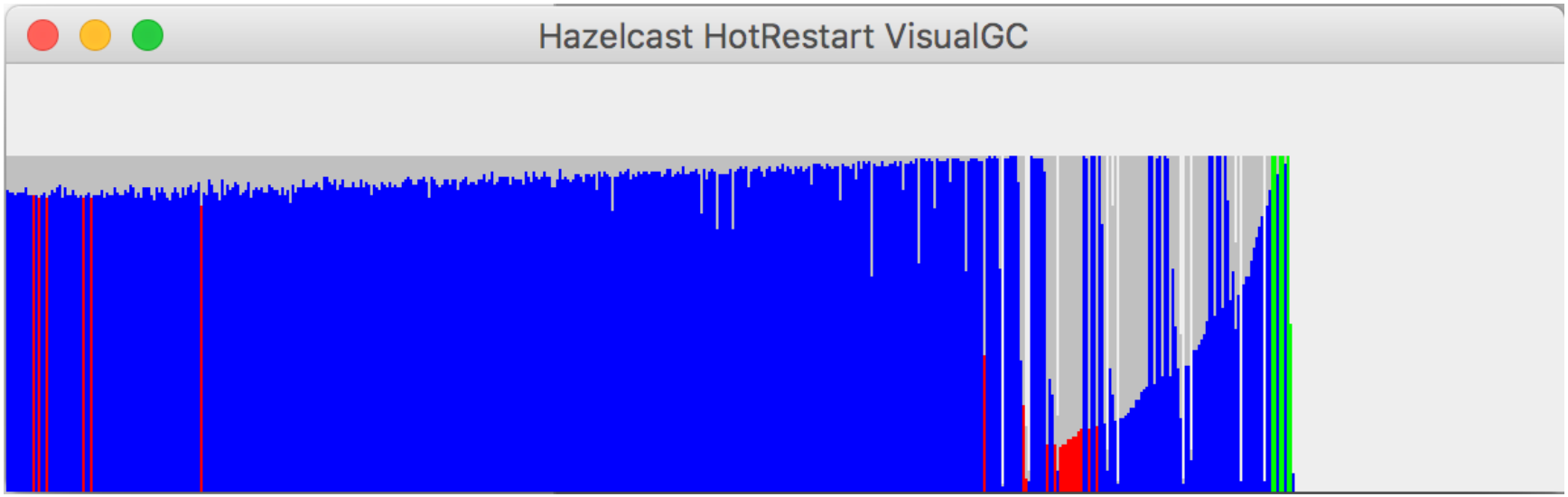
Design details: reloading

- pipelined mulitithreaded design
- 1 I/O thread -> N partition threads -> 1 GC thread
- SSD can feed several concurrent input streams
  - *parallelism* option
  - combined throughput > single-threaded

Diagram: reloading



Live Demo



Factors affecting performance

- sensitive to entry size
- fixed cost per entry => small entries have less throughput
- more garbage, longer reloading times
- parallelism helps with small entries, involving more cores in parsing the input streams

Benchmarks: the hardware

- Performance measured in Hazelcast's lab and on standard EC2 instances
- Lab specs:
  - dual-socket Xeon, 10 cores per socket (20 cores total)
  - SSD: PCIe-connected, SATA protocol
    - sequential reading, 1 thread: 0.8 GB/s
    - sequential reading, 4 threads: 1.8 GB/s

Results in the lab

- with 1 KB entry size, 100 million entries, 40 partition threads, parallelism of 4:
- 365,000 update ops per second
- reloaded in 82 seconds (1.2 GB/s)
  - raw reading throughput: 1.4 GB/s

Results on R3.4xlarge instance

- 16 vCPUs, 320 GB instance store
  - use instance store, not EBS
- 1 KB entry size, 40 million entries, 16 partition threads, parallelism of 2:
  - 185,000 ops per second
  - 128 seconds to reload (312 MB/s)
  - raw reading throughput: 360 MB/s

Future use cases

- currently Hot Restart supports only atomic full-cluster shutdown and restart
- planned:
  - restart with missing members, restore data from backups
  - restart with partial data (still saves a lot of time)
  - use Hot Restart to migrate data to another cluster
    - e.g., from production to test environment

