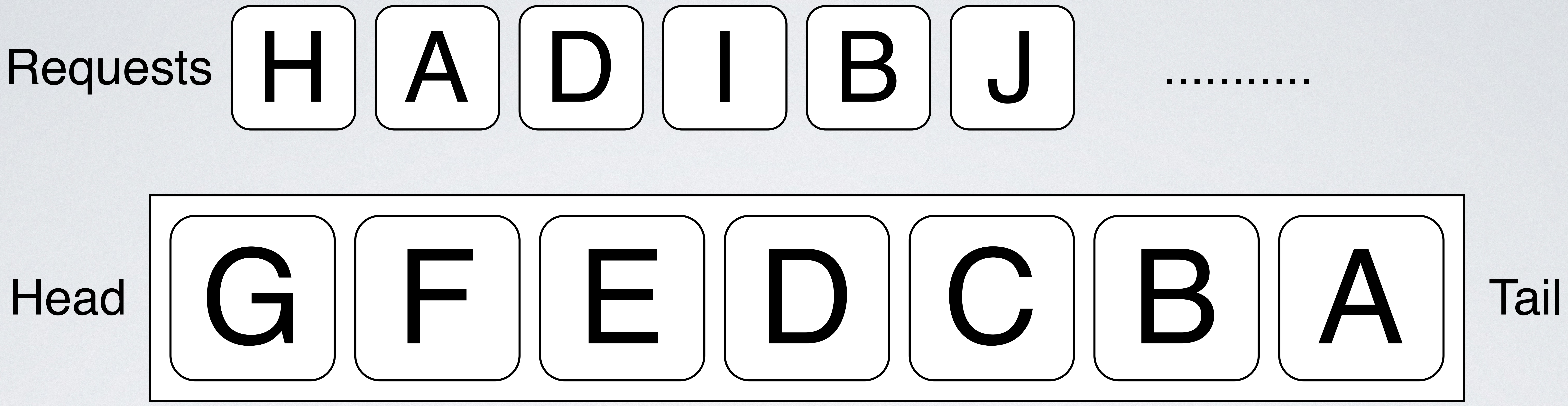
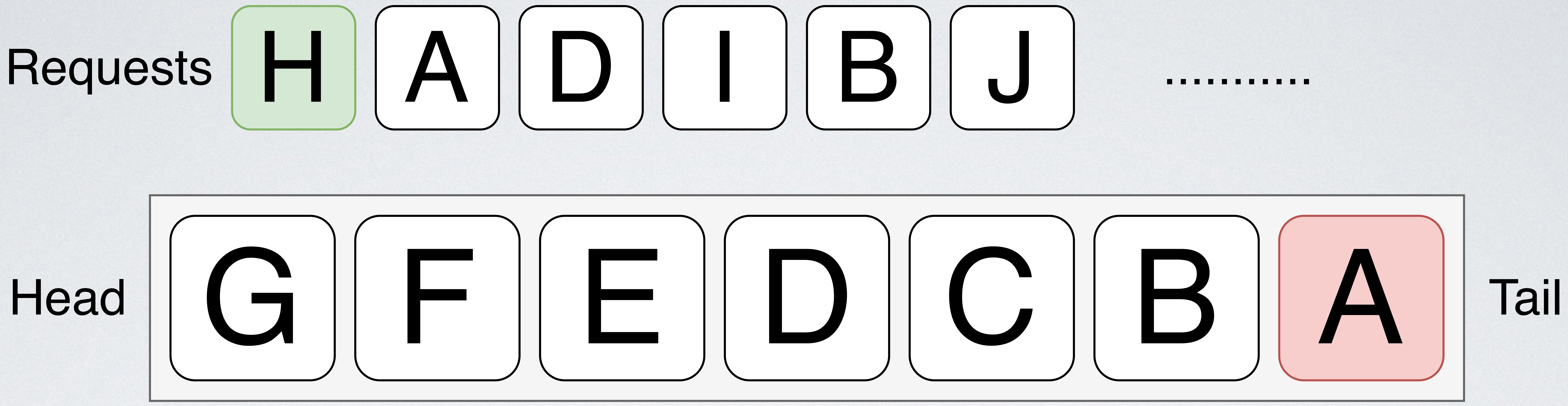
SIEVE

An Eviction Algorithm Simpler than LRU for ~~Web~~ Caches

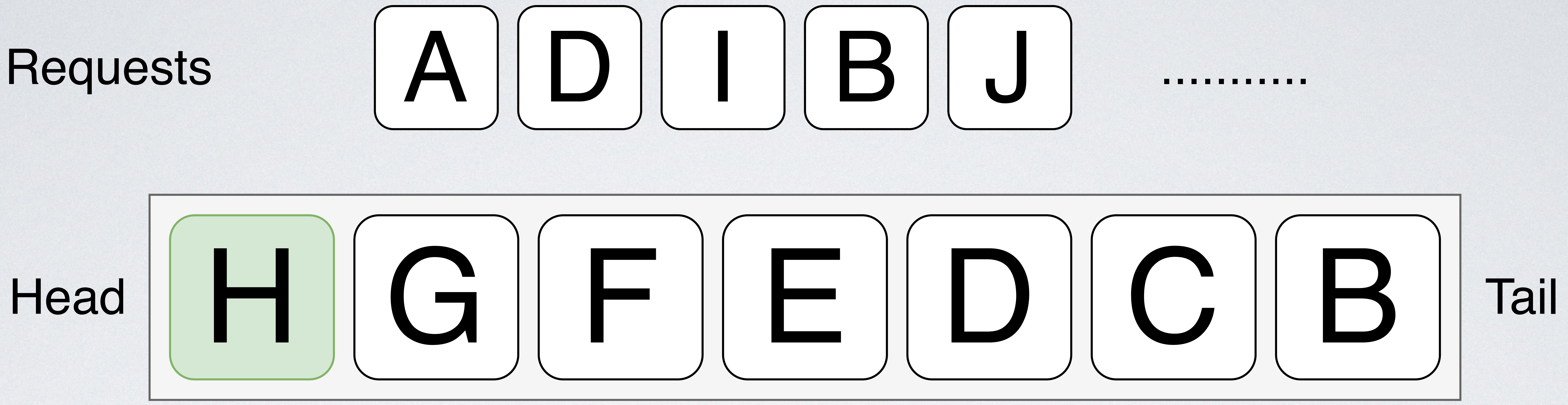
Ondřej Surý, ISC; IEPG Bangkok; 2025-03-16



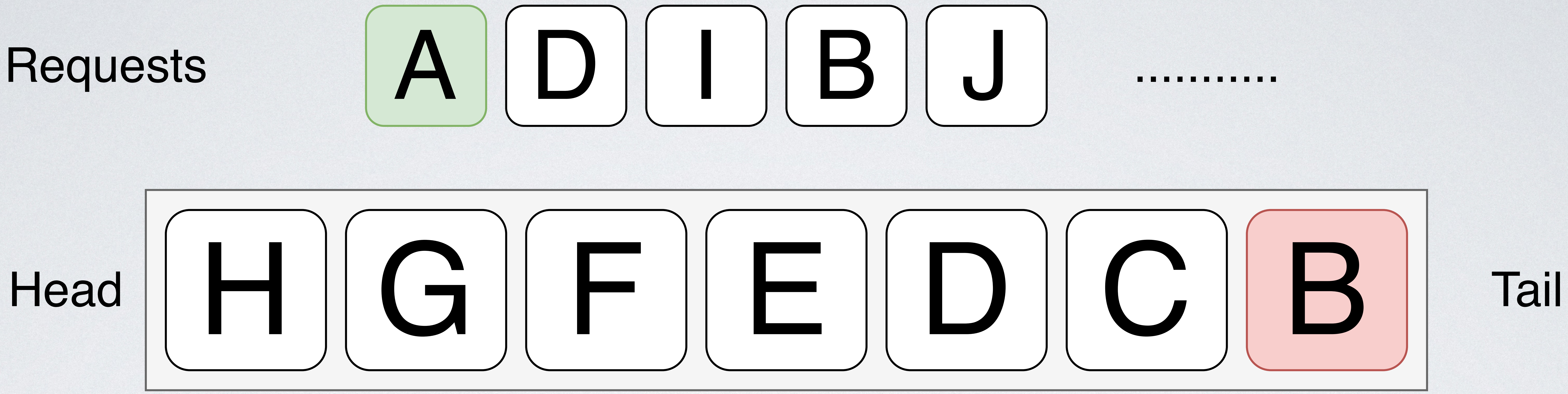
HOW DOES LRU WORK



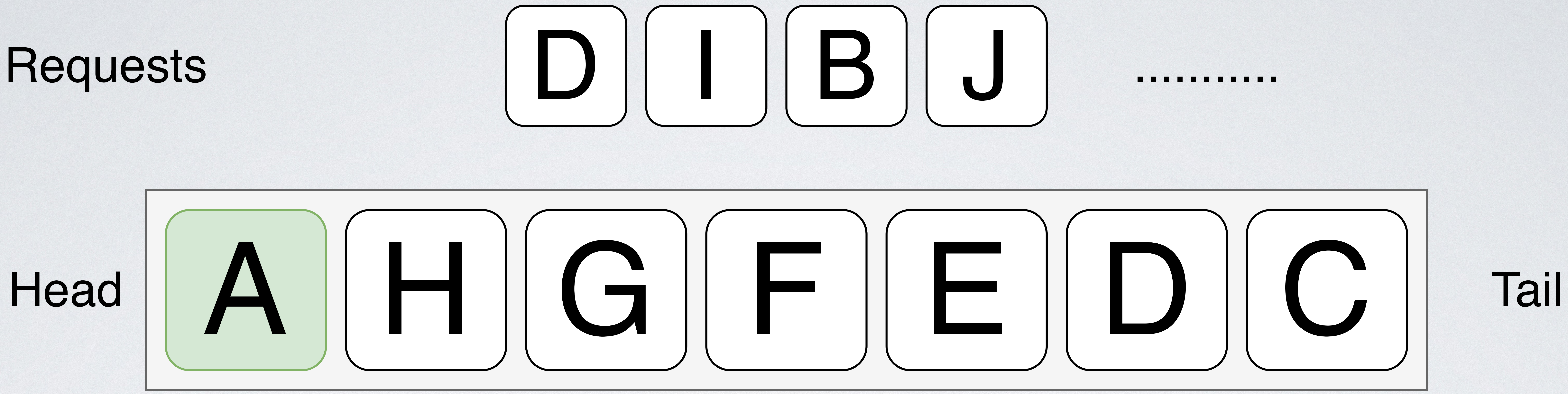
HOW DOES LRU WORK



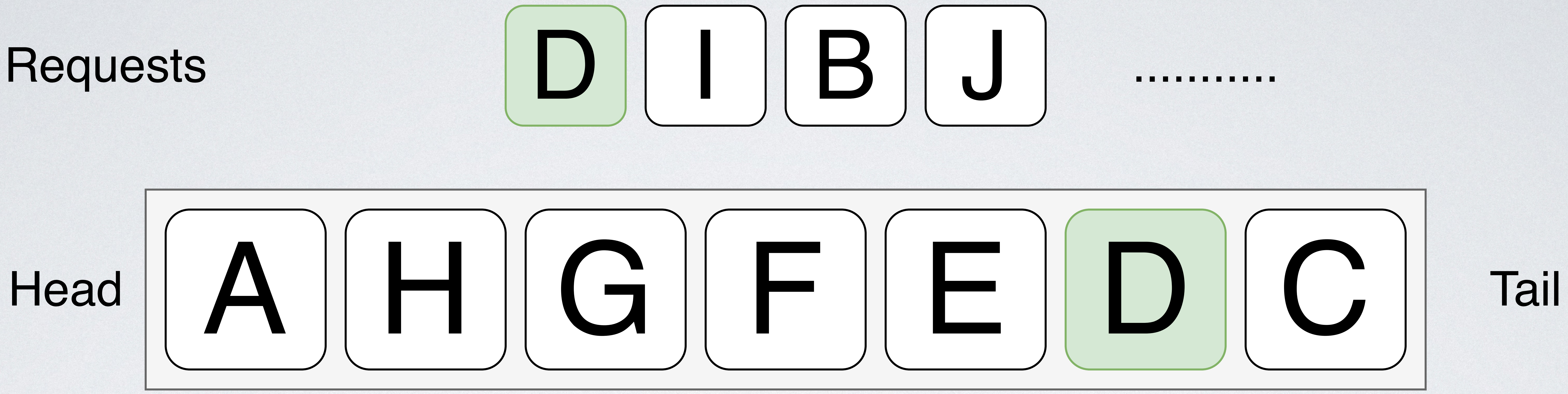
HOW DOES LRU WORK



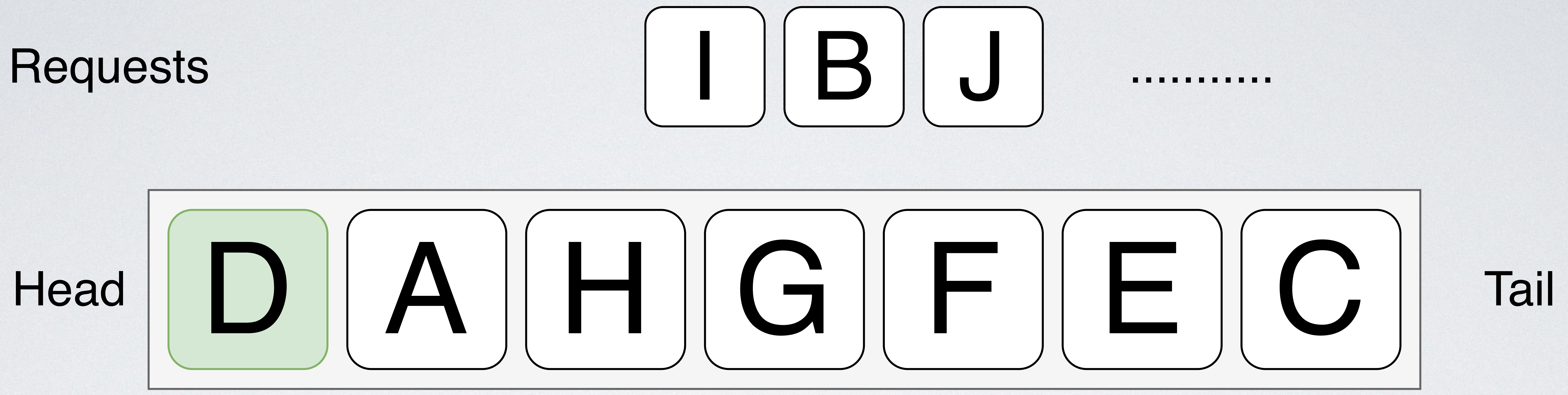
HOW DOES LRU WORK



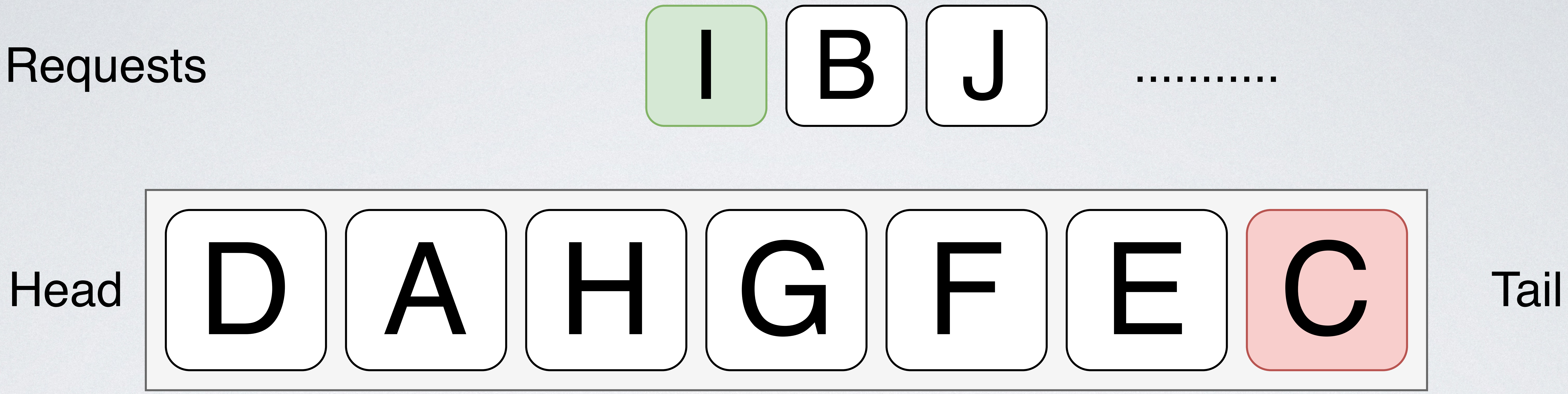
HOW DOES LRU WORK



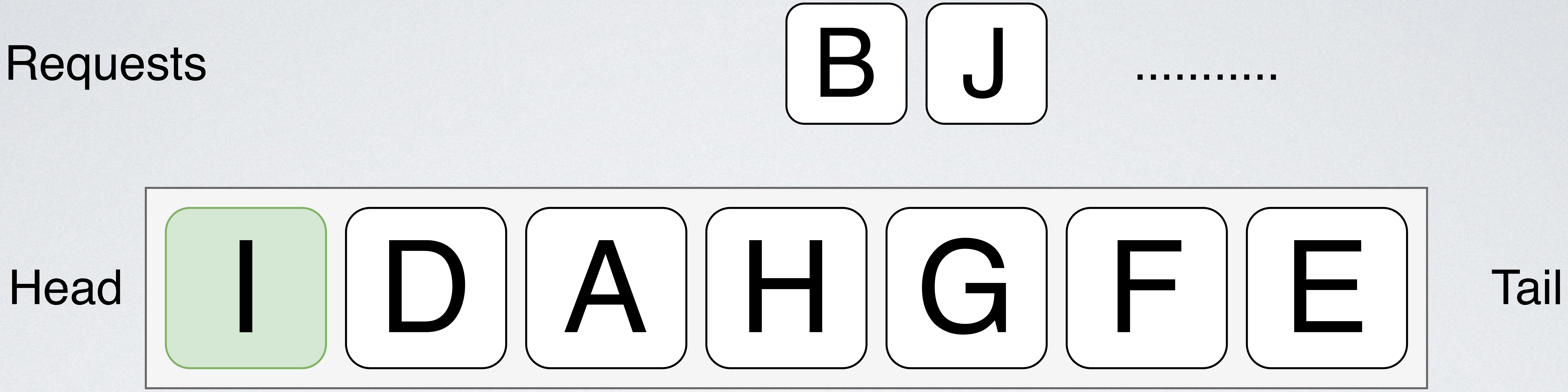
HOW DOES LRU WORK



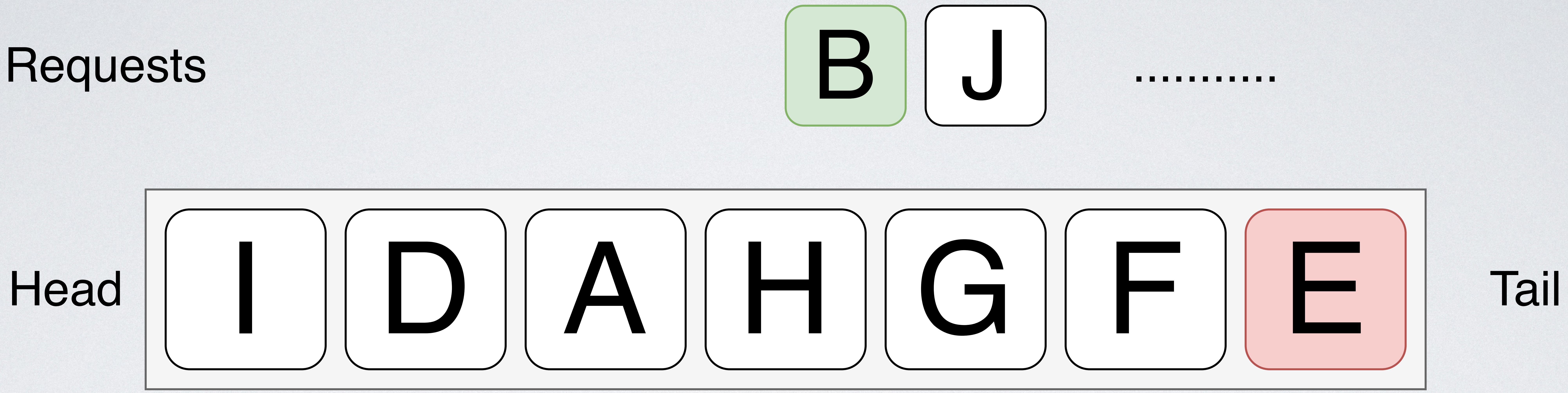
HOW DOES LRU WORK



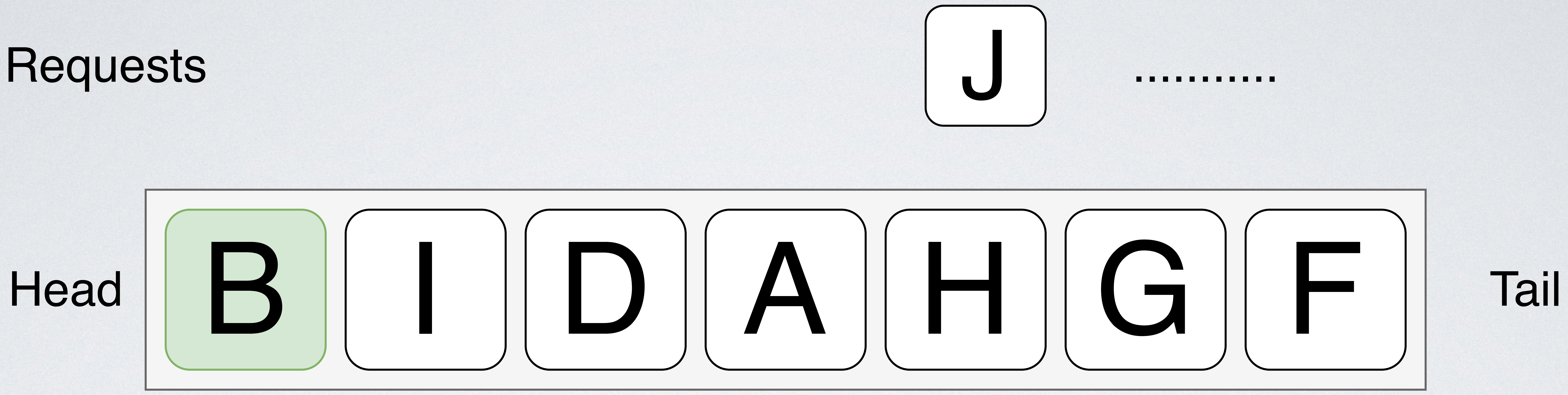
HOW DOES LRU WORK



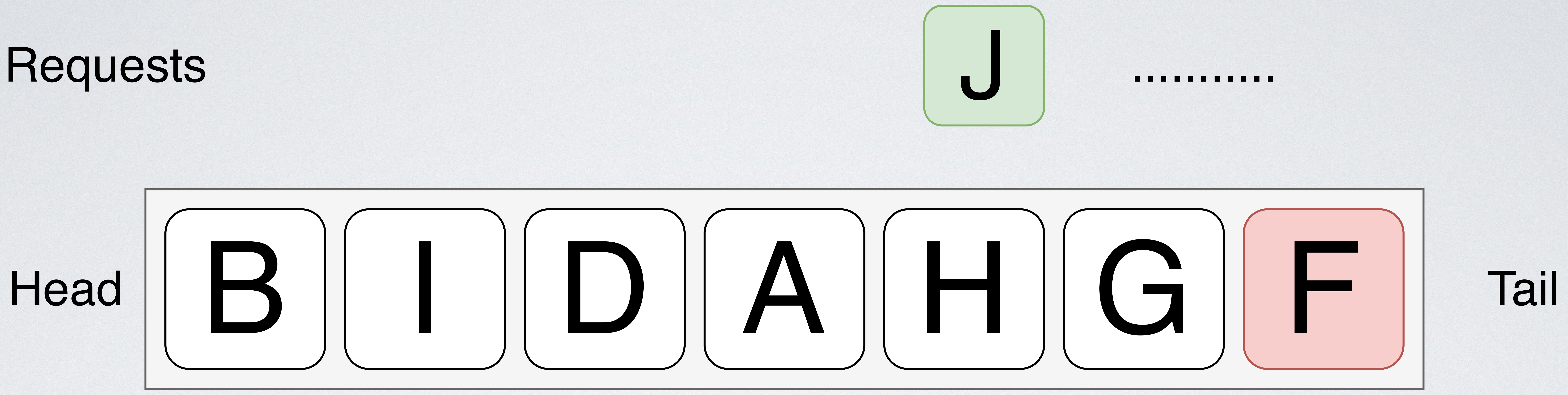
HOW DOES LRU WORK



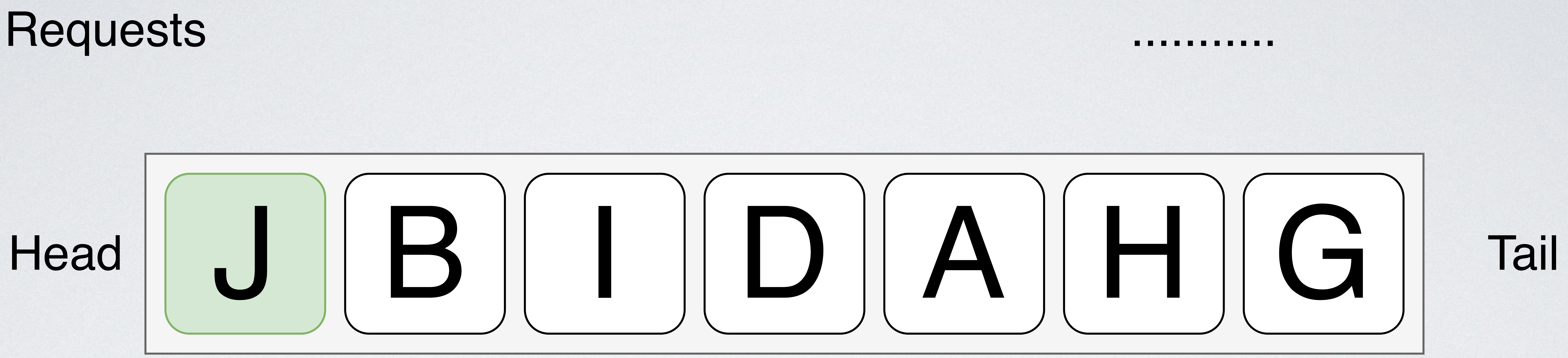
HOW DOES LRU WORK



HOW DOES LRU WORK



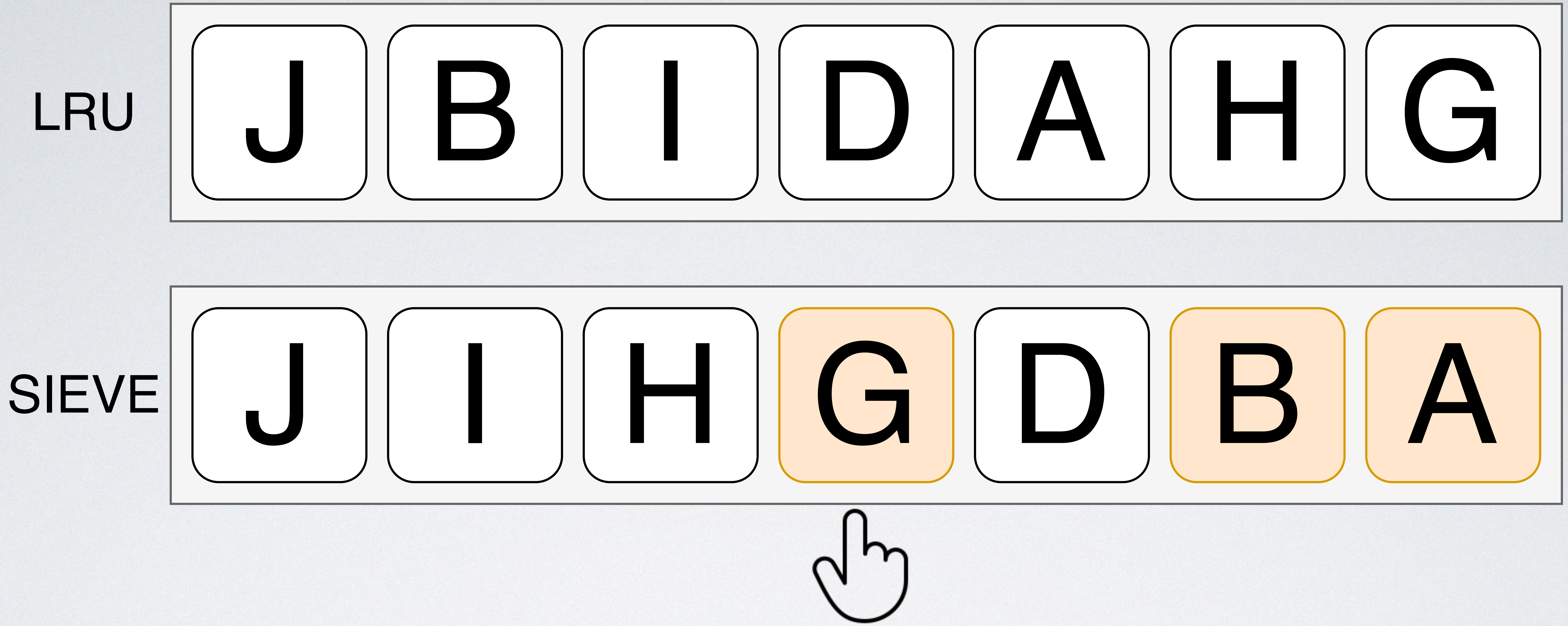
HOW DOES LRU WORK



HOW DOES LRU WORK

An Example of SIEVE

HOW DOES SIEVE WORK?



LRU VS SIEVE

SIEVE VS LRU

- CACHE HIT
  - LRU:
    - Moves object to the HEAD
    - Locks the list
  - SIEVE:
    - Mark object (as VISITED)
    - No list-level locking

- CACHE MISS
  - BOTH
    - Insert new item at the list HEAD (both)
    - Locks the list
  - LRU:
    - Evict item from the TAIL
  - SIEVE:
    - If marked, unmark and move the hand
    - If unmarked, evict

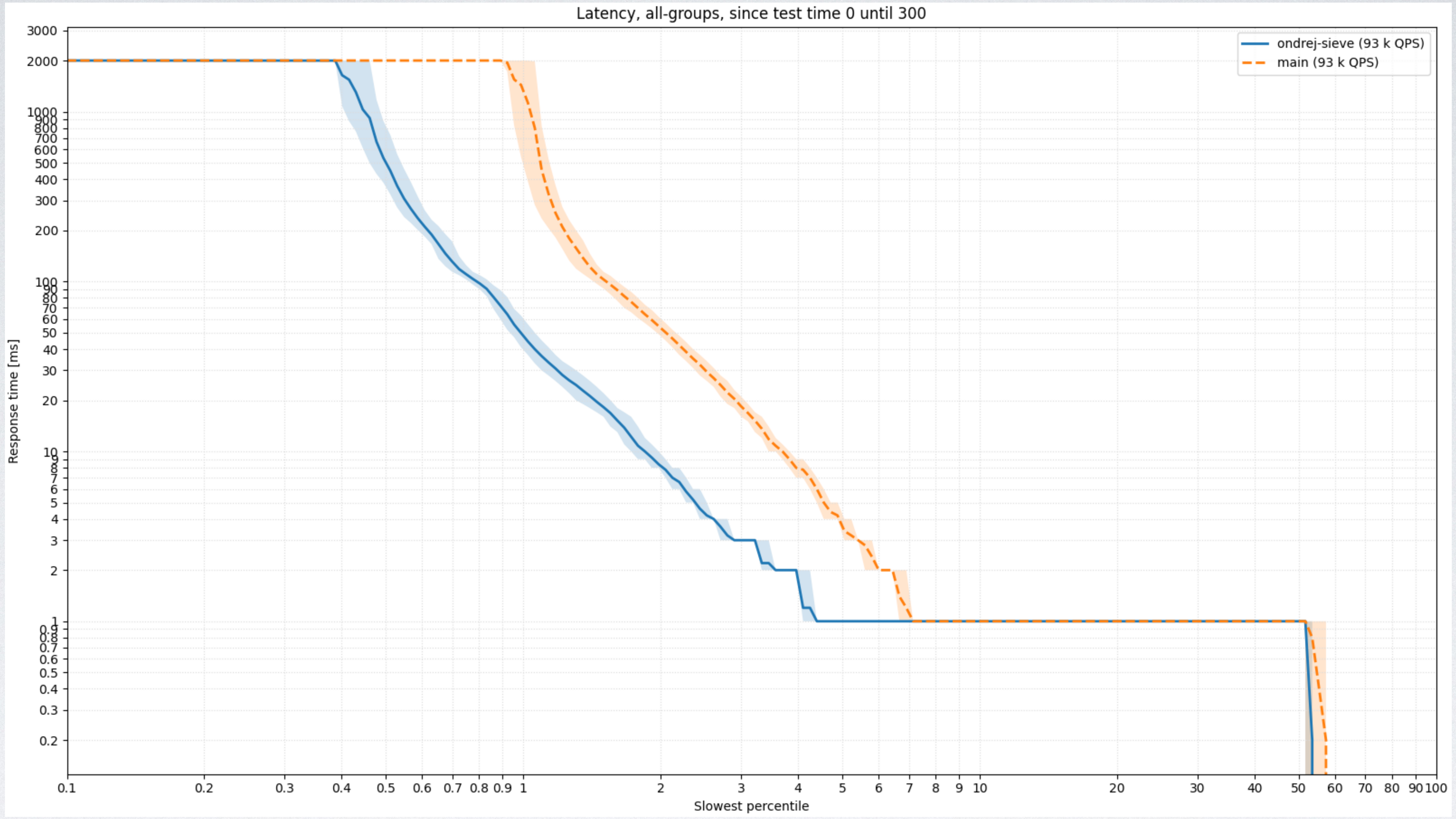
LAZY PROMOTION AND QUICK DEMOTION

- LAZY PROMOTION
  - Only promote the cached objects at the eviction time
- QUICK DEMOTION
  - Remove most objects quickly after insertion

SIEVE IN BIND 9

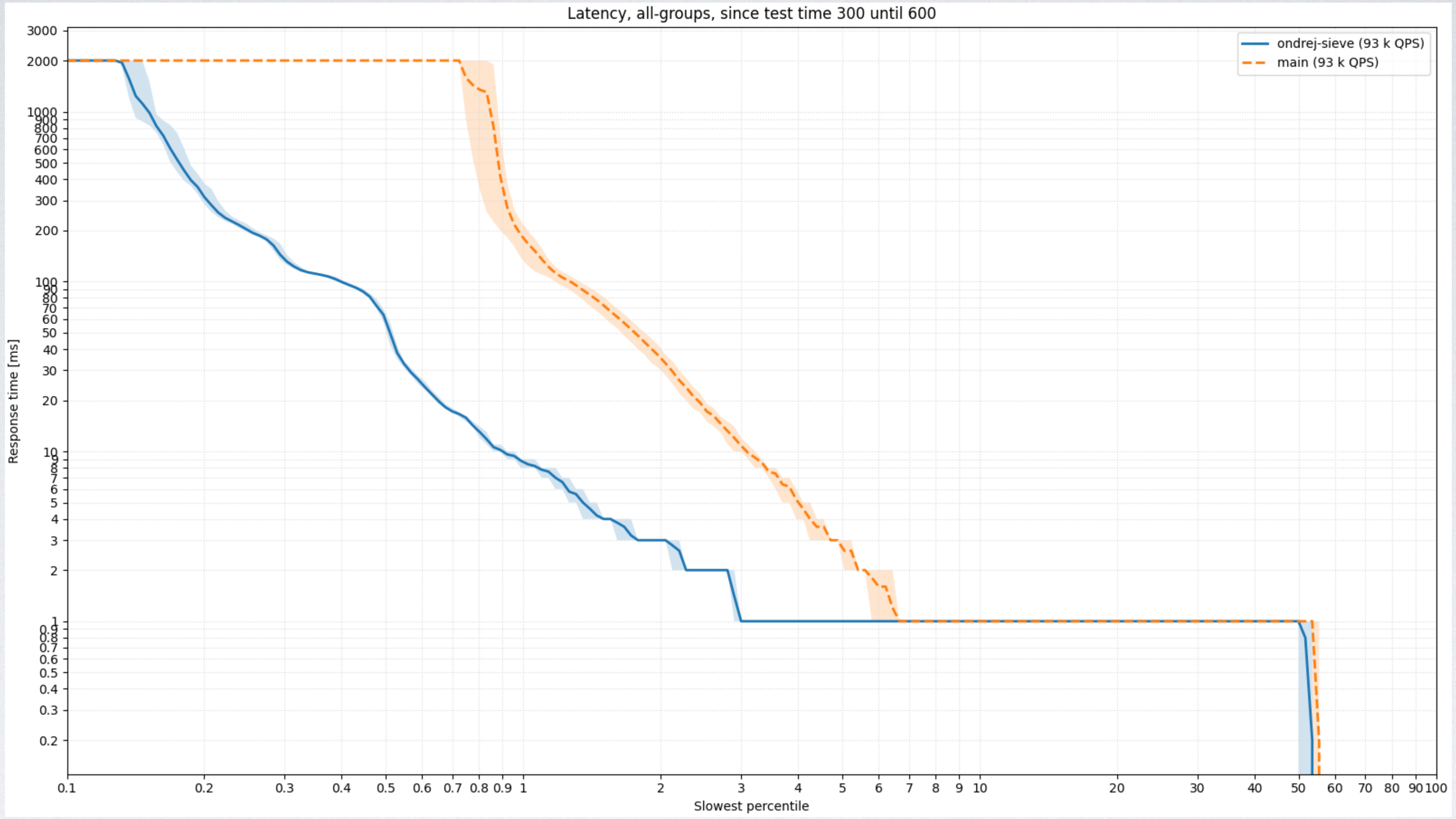
- Replace TTL-based and LRU-based cleaning with SIEVE
- The SIEVE implementation is simpler than the LRU implementation

```
$ git diff --stat origin/main
 lib/dns/include/dns/rdataslab.h |  10 +--
 lib/dns/qpcache.c               | 512 ++++++++++++++++++++
+------------------------------------------------------------------
-----------------------------------------------------------
 lib/isc/Makefile.am             |   1 +
 lib/isc/include/isc/sieve.h     |  62 ++++++++++++++++++
 4 files changed, 144 insertions(+), 441 deletions(-)
```



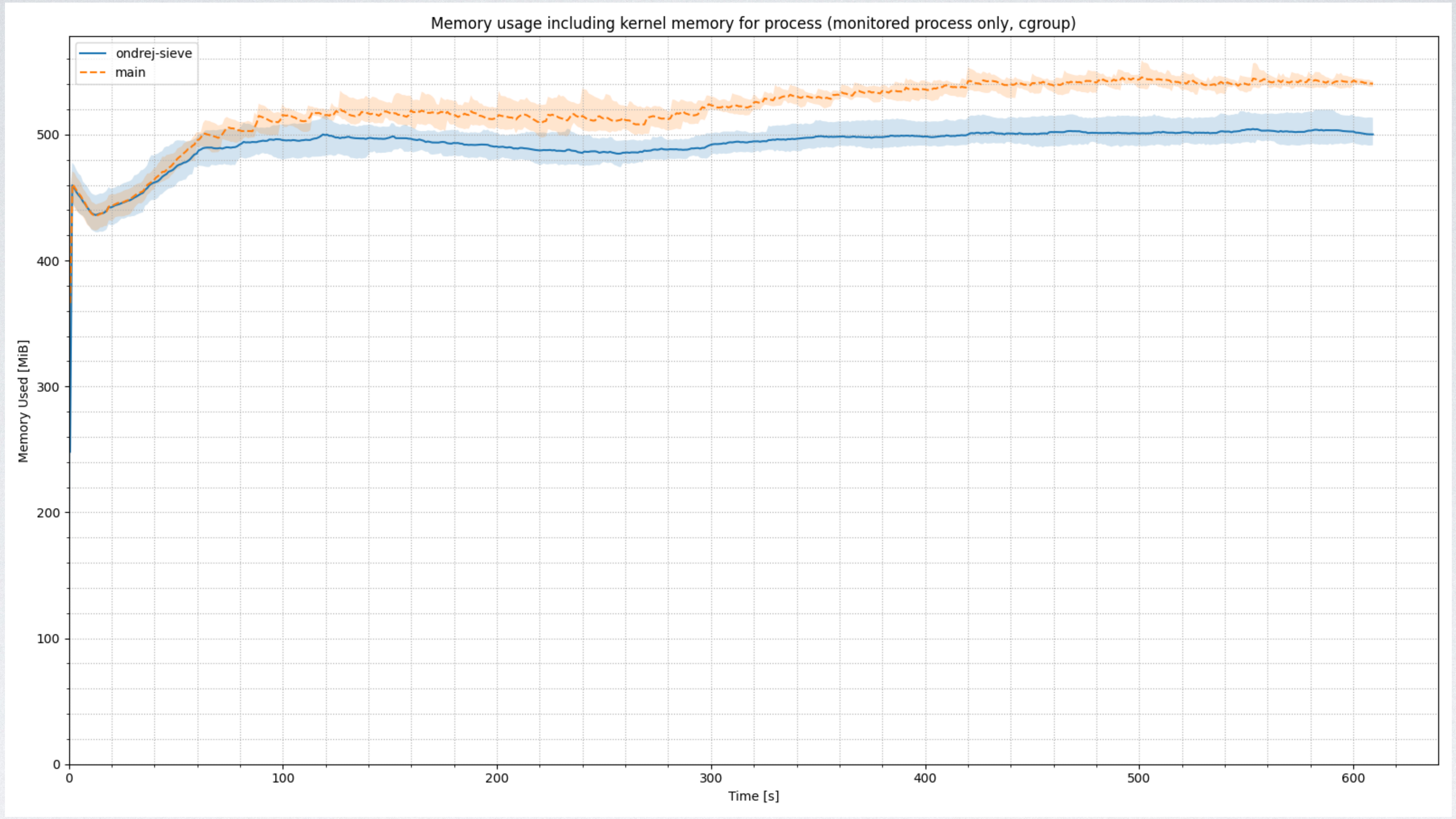
BIND 9 (128MB CACHE)

Latency, Cold-Cache



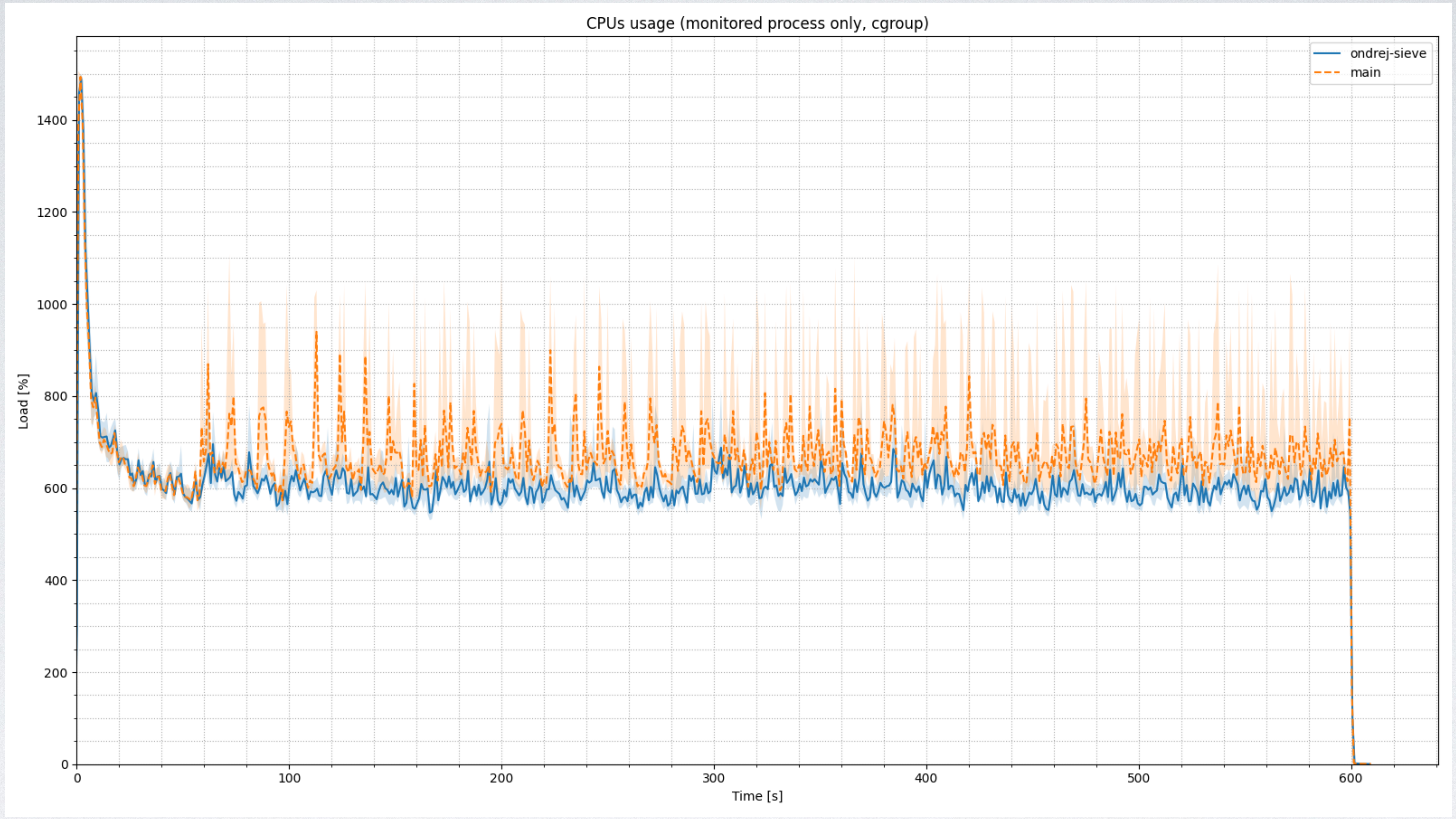
BIND 9 (128MB CACHE)

Latency, Hot-Cache



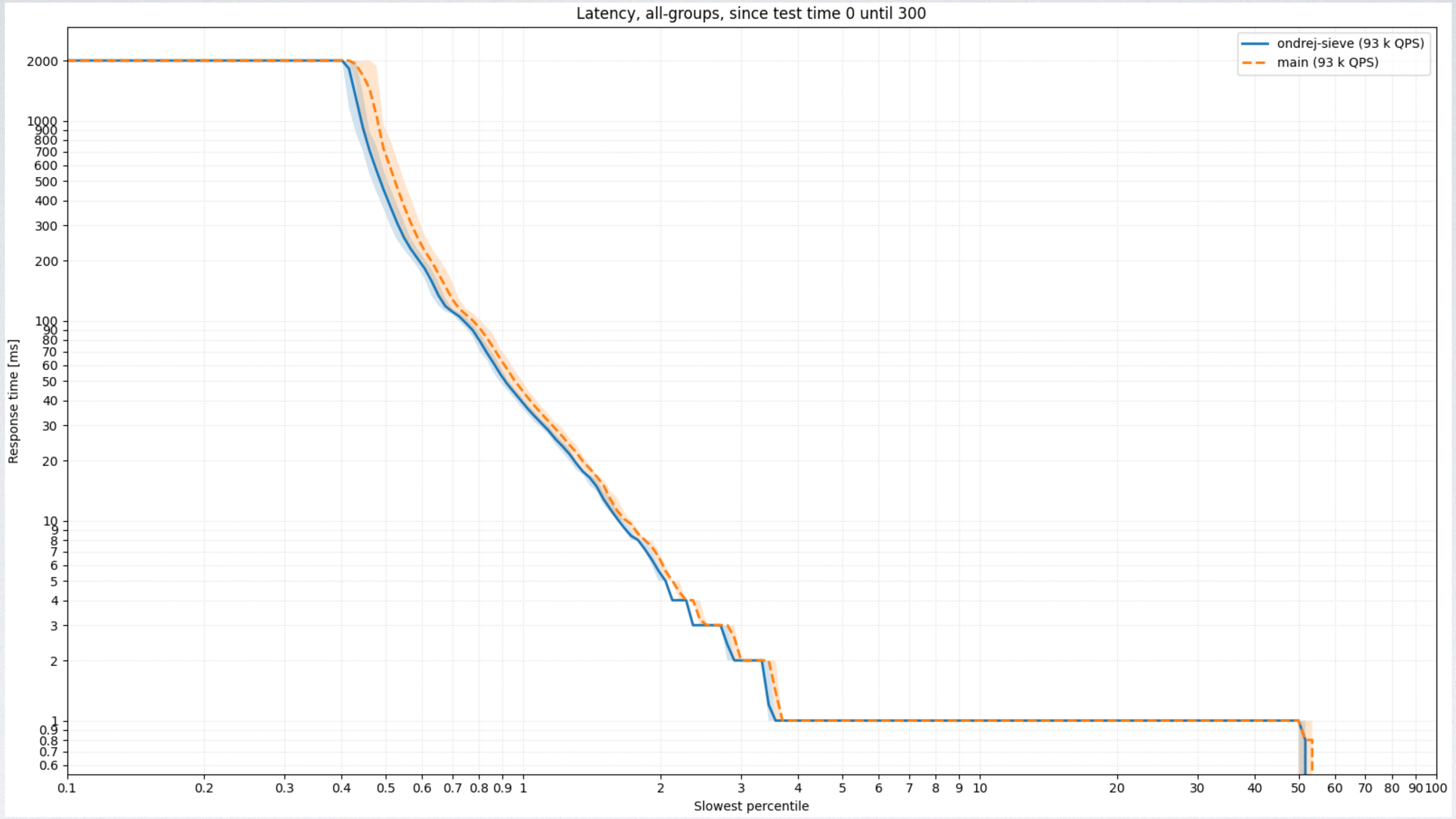
BIND 9 (128MB CACHE)

Memory



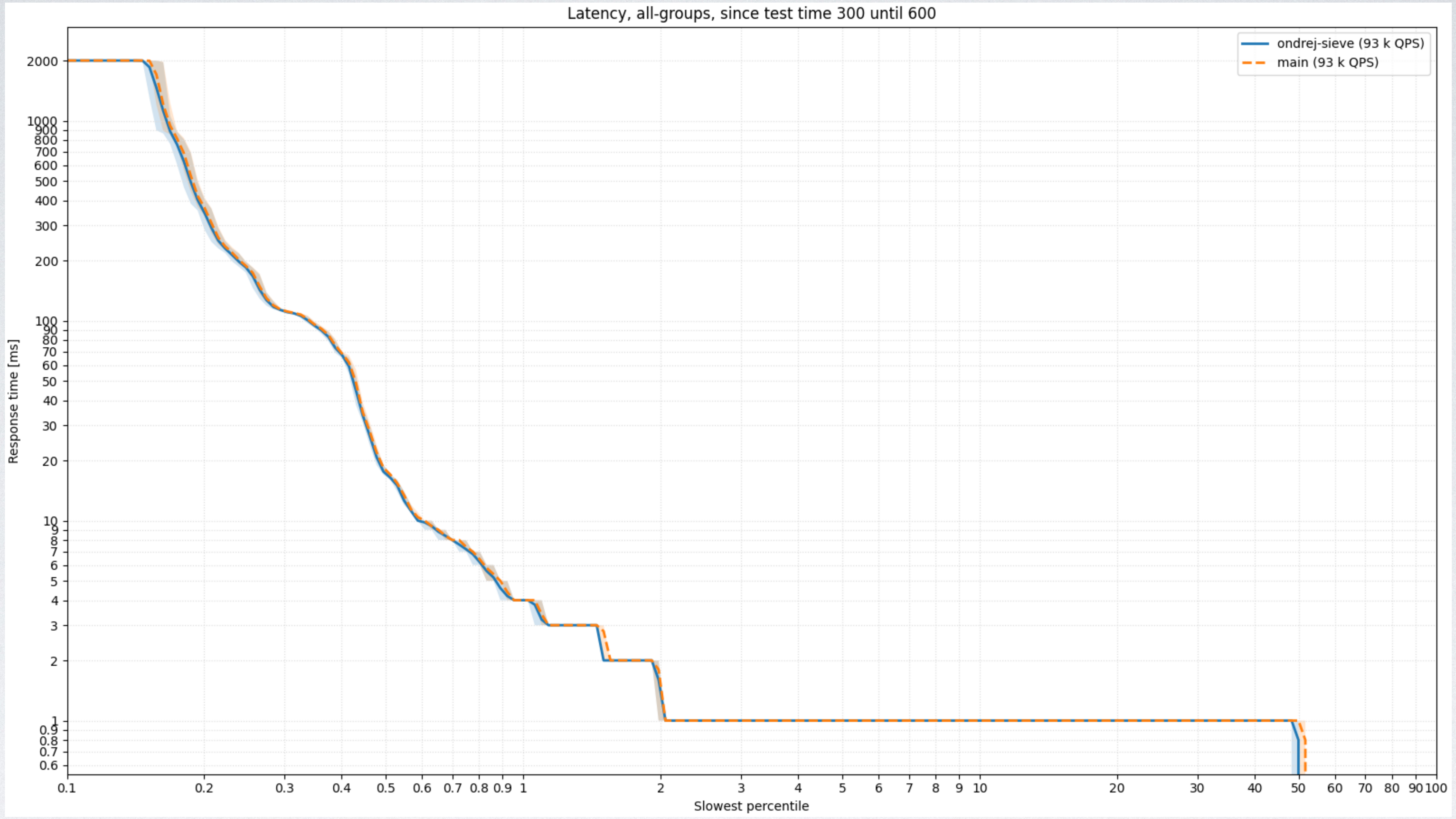
BIND 9 (128MB CACHE)

CPU



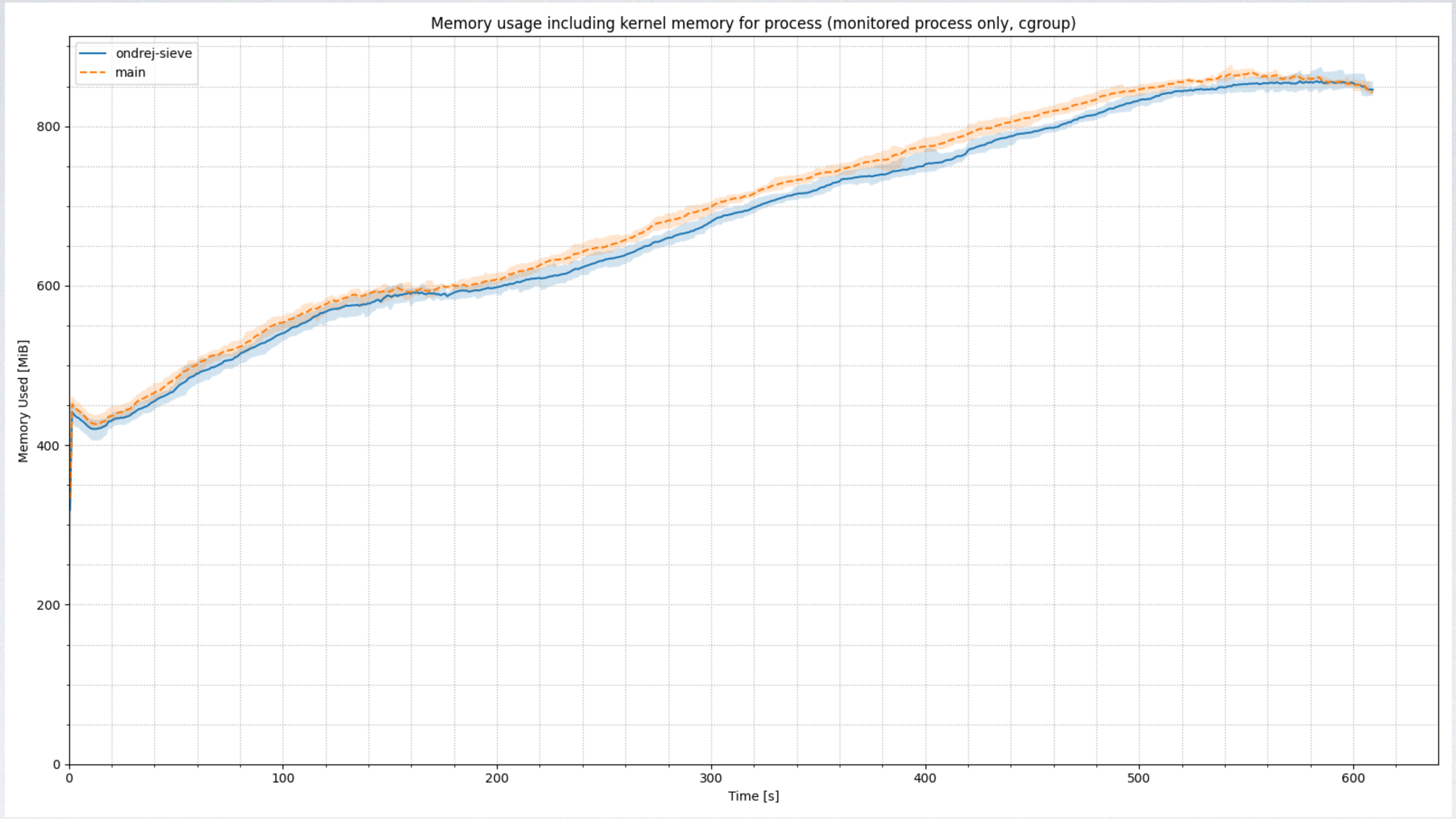
BIND 9 (30GB CACHE)

Latency, Cold-Cache



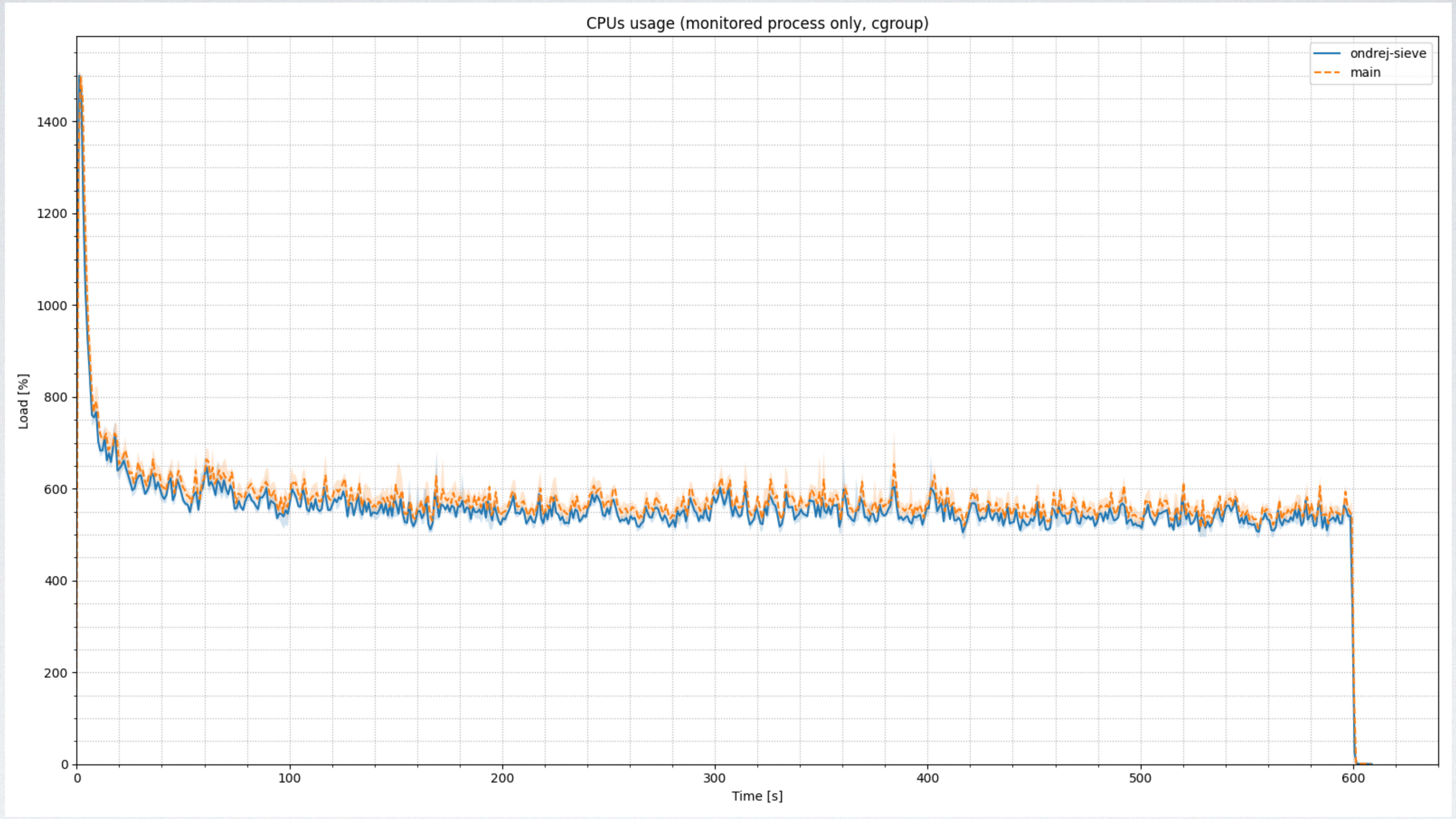
BIND 9 (30GB CACHE)

Latency, Hot-Cache



BIND 9 (30GB CACHE)

Memory



BIND 9 (30GB CACHE)

Memory

SIEVE IN BIND 9

- Implementation in BIND 9.21+ (development version)
- Do we keep TTL-based cleaning?
  - Probably not
- How does the algorithm behave with large caches?
  - More research needed (for BIND 9 and for my dissertation)

THANK YOU

REFERENCES

- https://sievecache.com/
- https://junchengyang.com/publication/nsdi24-SIEVE.pdf
- https://cachemon.github.io/SIEVE-website/blog/2023/12/17/sieve-is-simpler-than-lru/
- https://isc-projects.gitlab-pages.isc.org/-/bind9-shotgun-ci/-/jobs/5392303/artifacts/index.html (30 GB cache)
- https://isc-projects.gitlab-pages.isc.org/-/bind9-shotgun-ci/-/jobs/5392288/artifacts/index.html (128 MB cache)