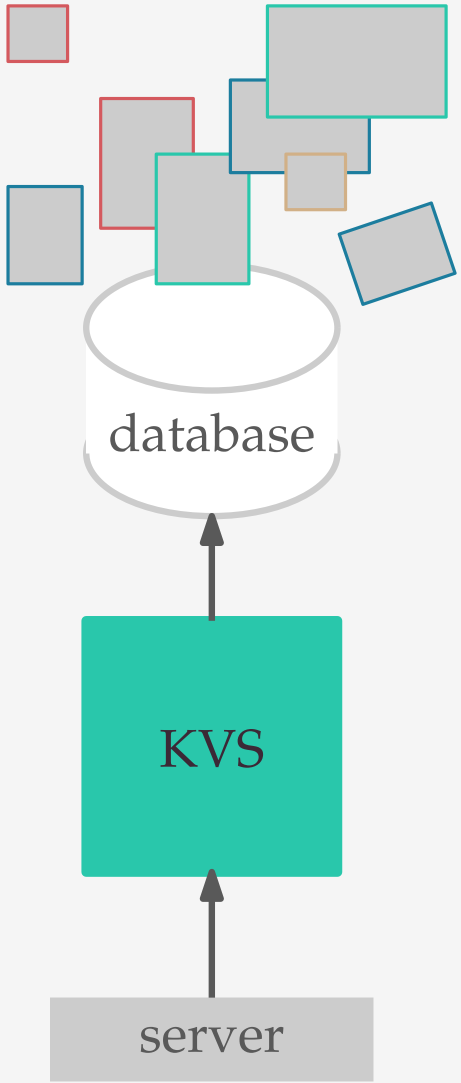
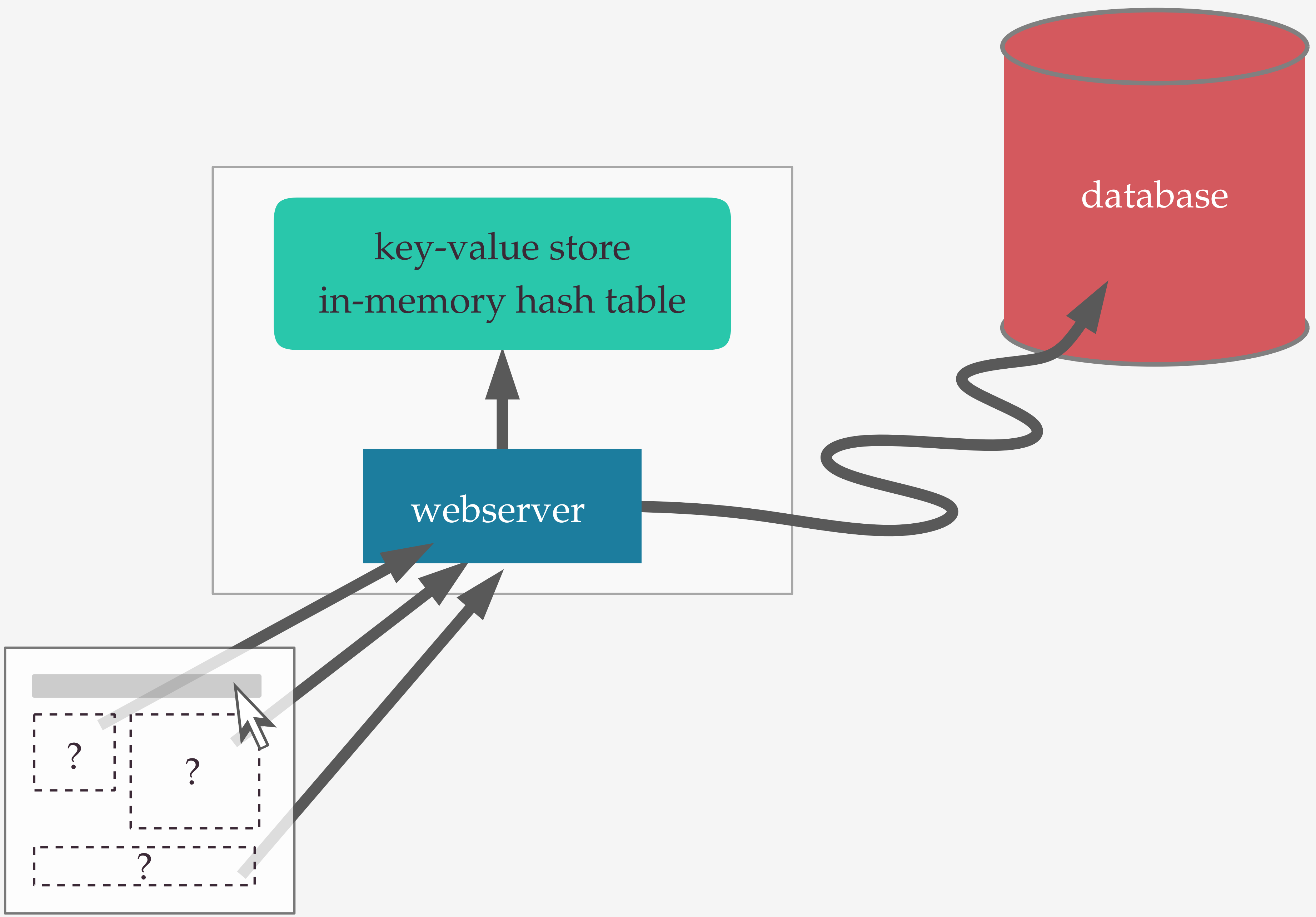
CACHE OPTIMIZATION FOR THE MODERN WEB

Jenny Lam

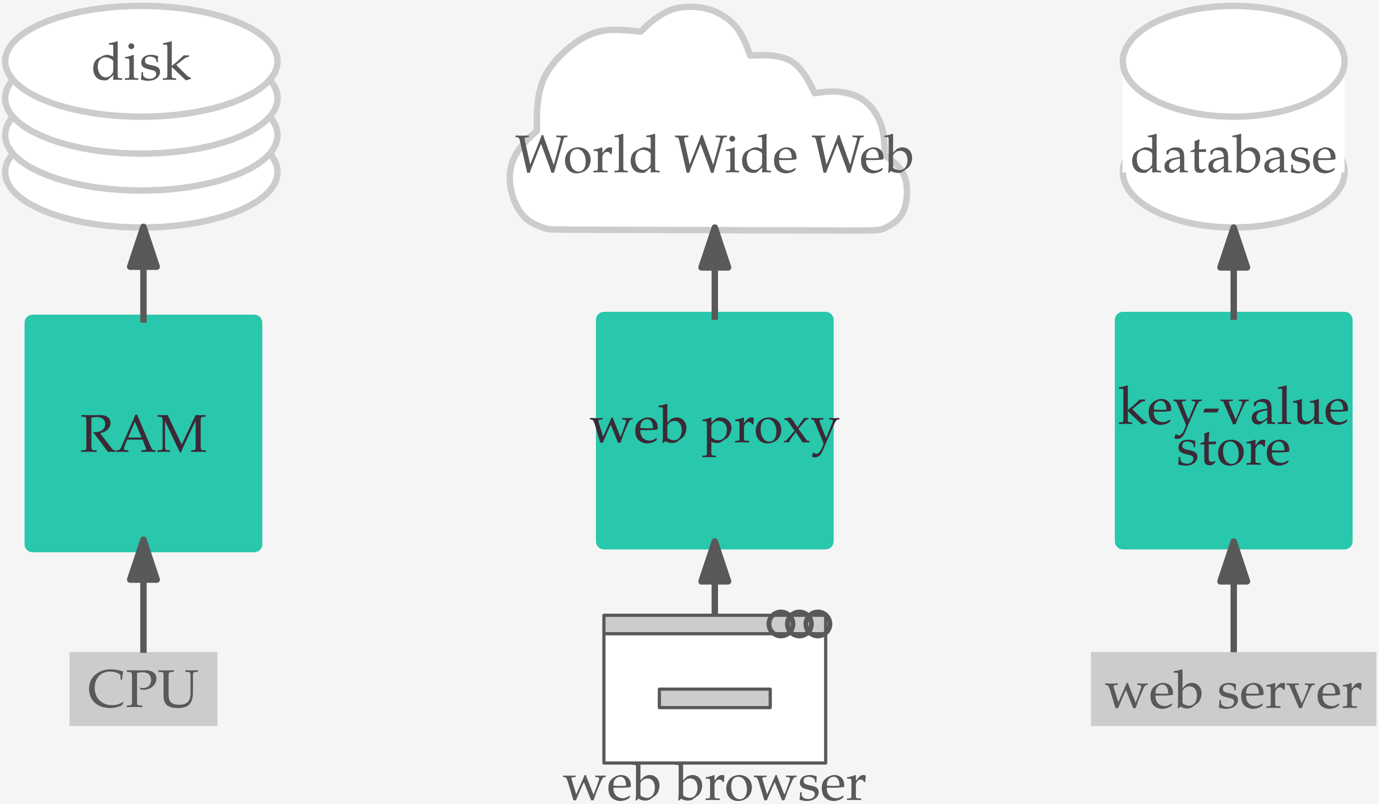
JOINT WORK WITH

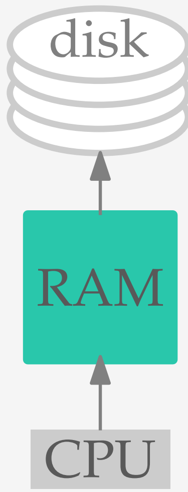
Shahram Ghandeharizadeh
Sandy Irani
Jason Yap

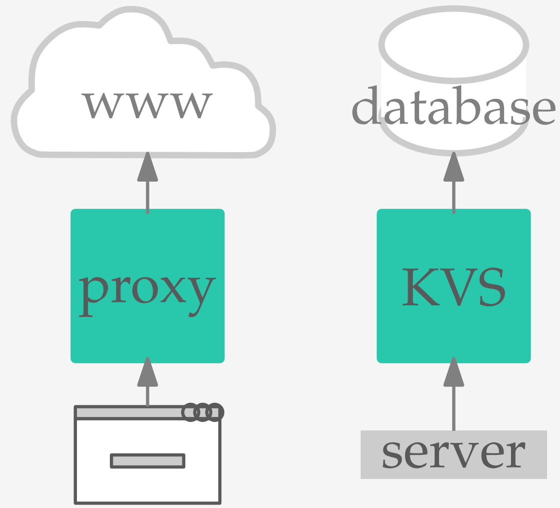


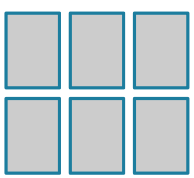


Scaling Memcache at Facebook, Nishtala et al., NSDI 2013.


disk
RAM
CPU
World Wide Web
web proxy
web browser
database
key-value
store
web server



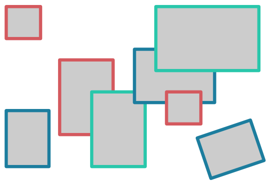




PAGING

minimize

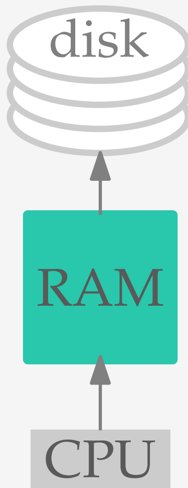
number of cache misses

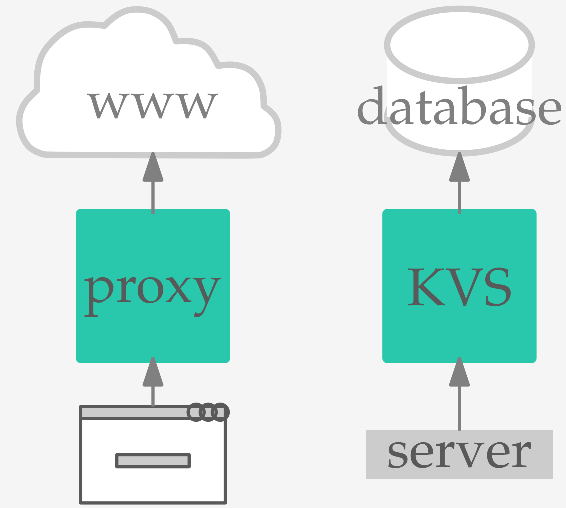


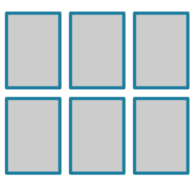
GENERALIZED CACHING

minimize

total cost of cache misses





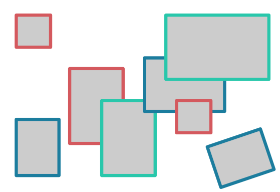


PAGING

minimize

number of cache misses

Least Recently Used (LRU)

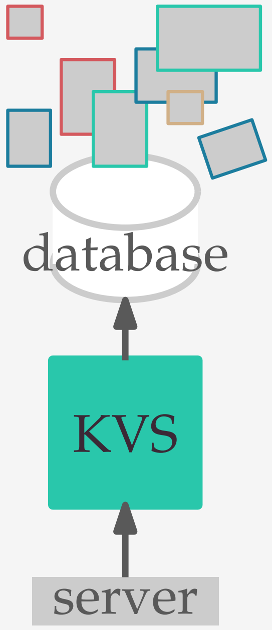


GENERALIZED CACHING

minimize

total cost of cache misses

GreedyDual-Size (GDS)



EVICTION POLICY

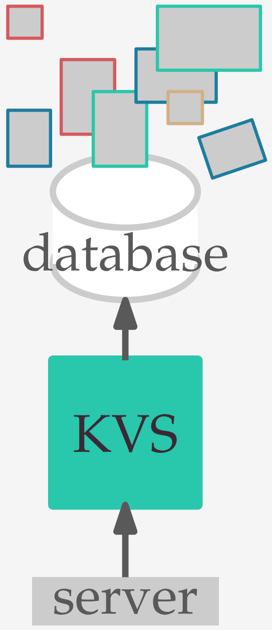
GDS → CAMP

PLACEMENT POLICY

generalized caching → managed memory caching

MEMORY HIERARCHY

2-level cache → multi-level cache



EVICTION POLICY

GDS → CAMP

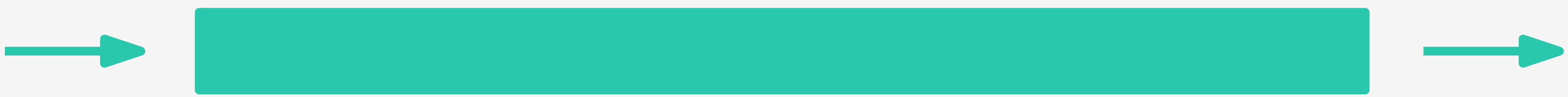
PLACEMENT POLICY

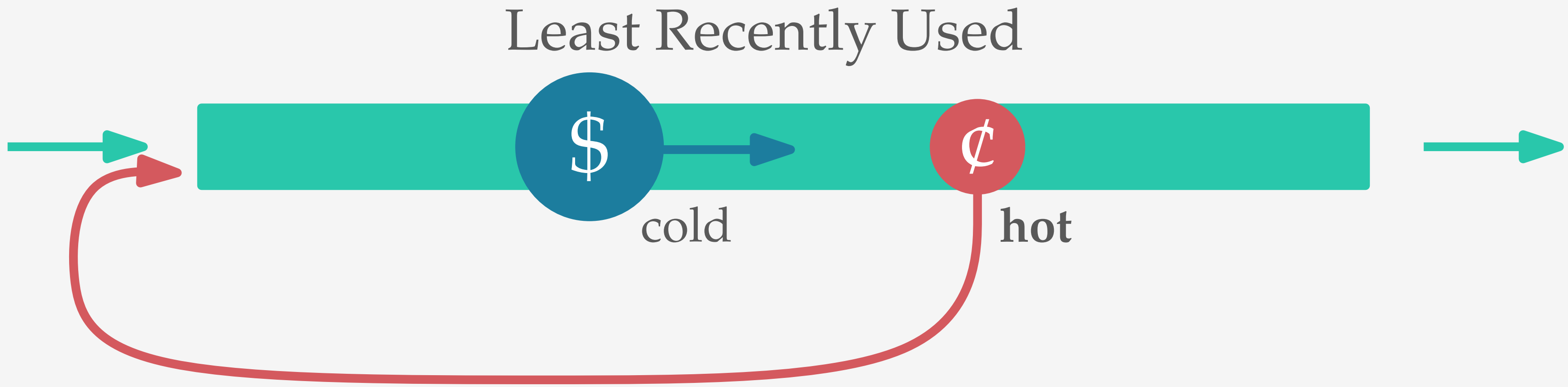
generalized caching → managed memory caching

MEMORY HIERARCHY

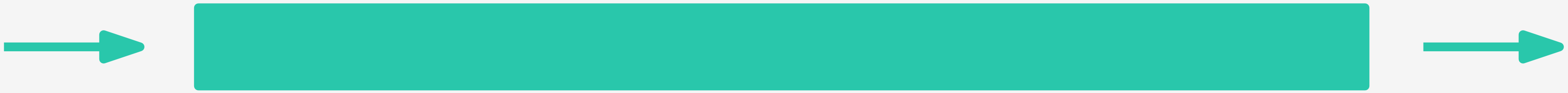
2-level cache → multi-level cache

Least Recently Used

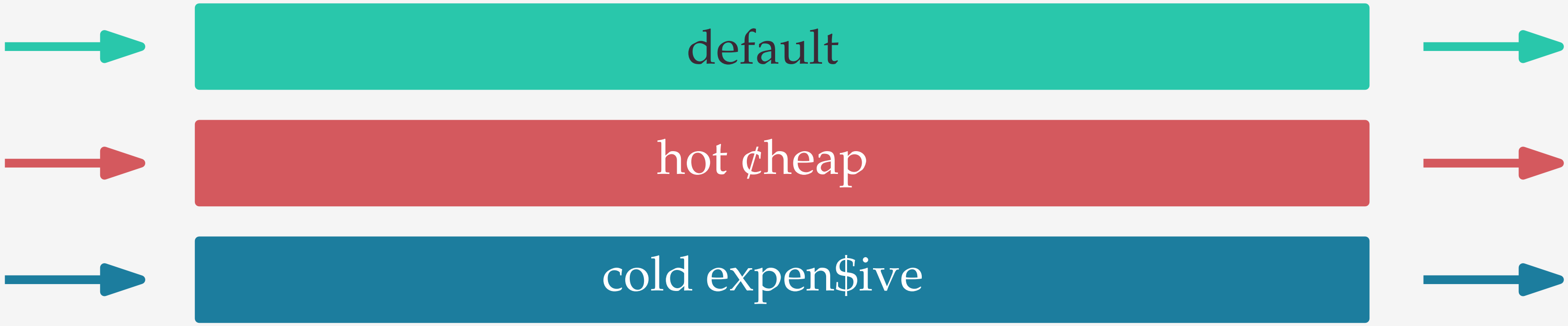




Least Recently Used
$
cold
¢
hot

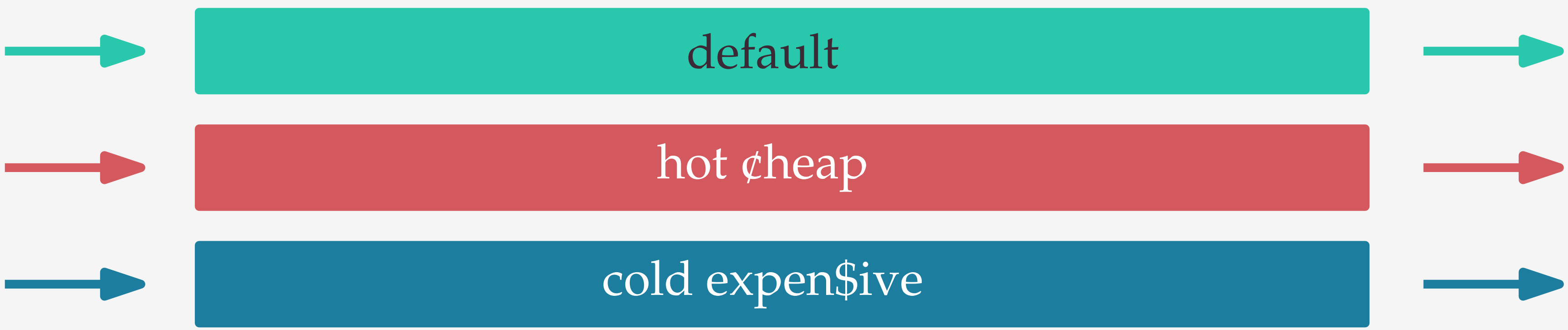


pooled Least Recently Used



Scaling Memcache at Facebook, Nishtala et al., NSDI 2013.

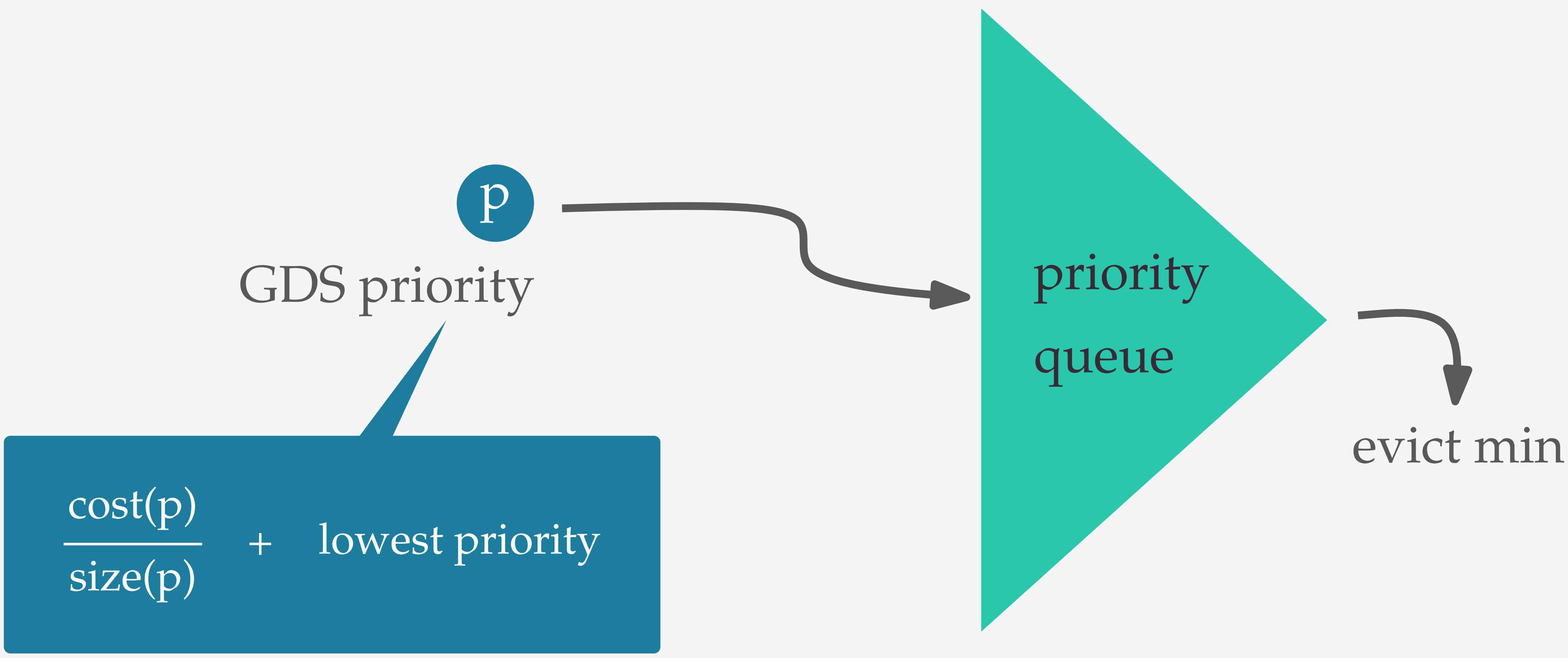
pooled Least Recently Used



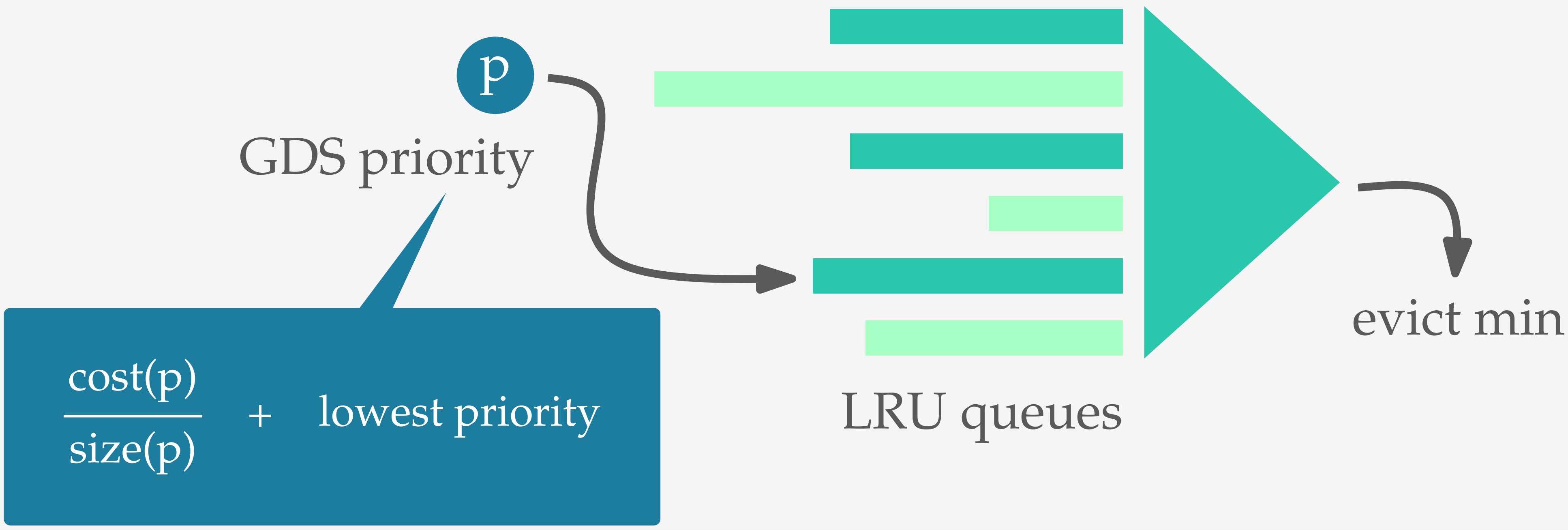
Scaling Memcache at Facebook, Nishtala et al., NSDI 2013.

need to take **recomputation cost** into consideration

GDS


p
GDS priority
$\frac{\text{cost(p)}}{\text{size(p)}}$ + lowest priority
priority queue
evict min

CAMP

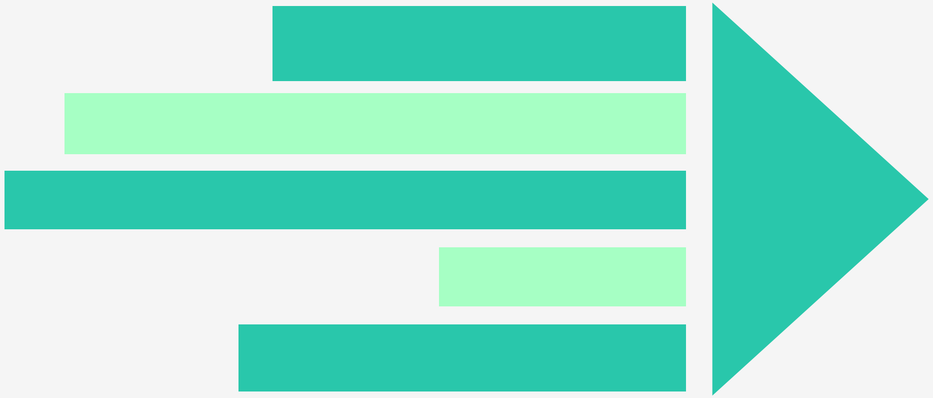


p
GDS priority
$\frac{\text{cost}(p)}{\text{size}(p)}$ + lowest priority
LRU queues
evict min

CAMP

$$\mathrm{round}\left(\frac{\mathrm{cost}(p)}{\mathrm{size}(p)}\right)$$

CAMP



$$\mathrm{round}\left(\frac{\mathrm{cost}(p)}{\mathrm{size}(p)}\right)$$

CAMP

$$\text{round}\left(\frac{\text{cost}(p)}{\text{size}(p)}\right)$$

PERFORMANCE

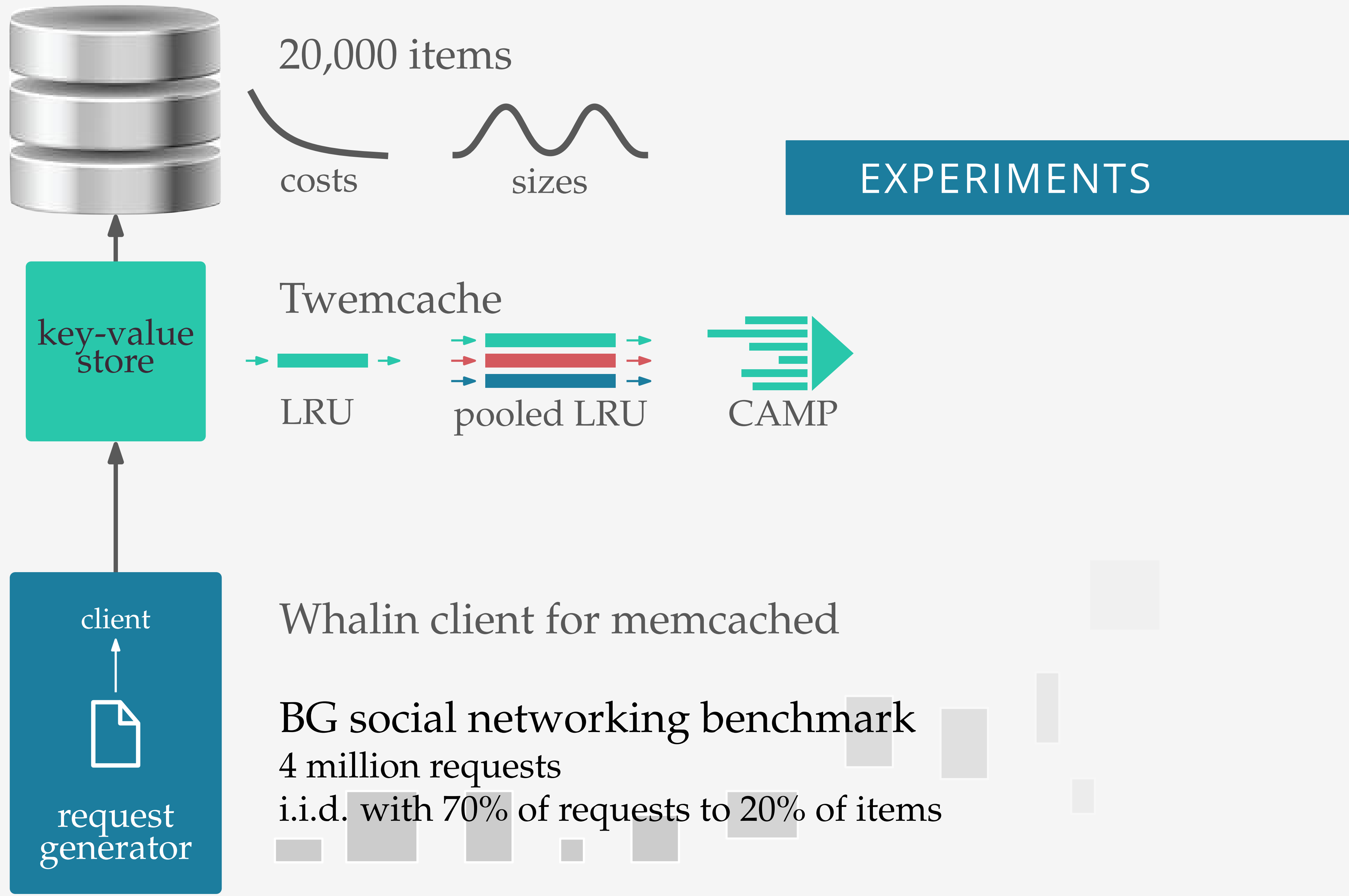
log (#items) per update

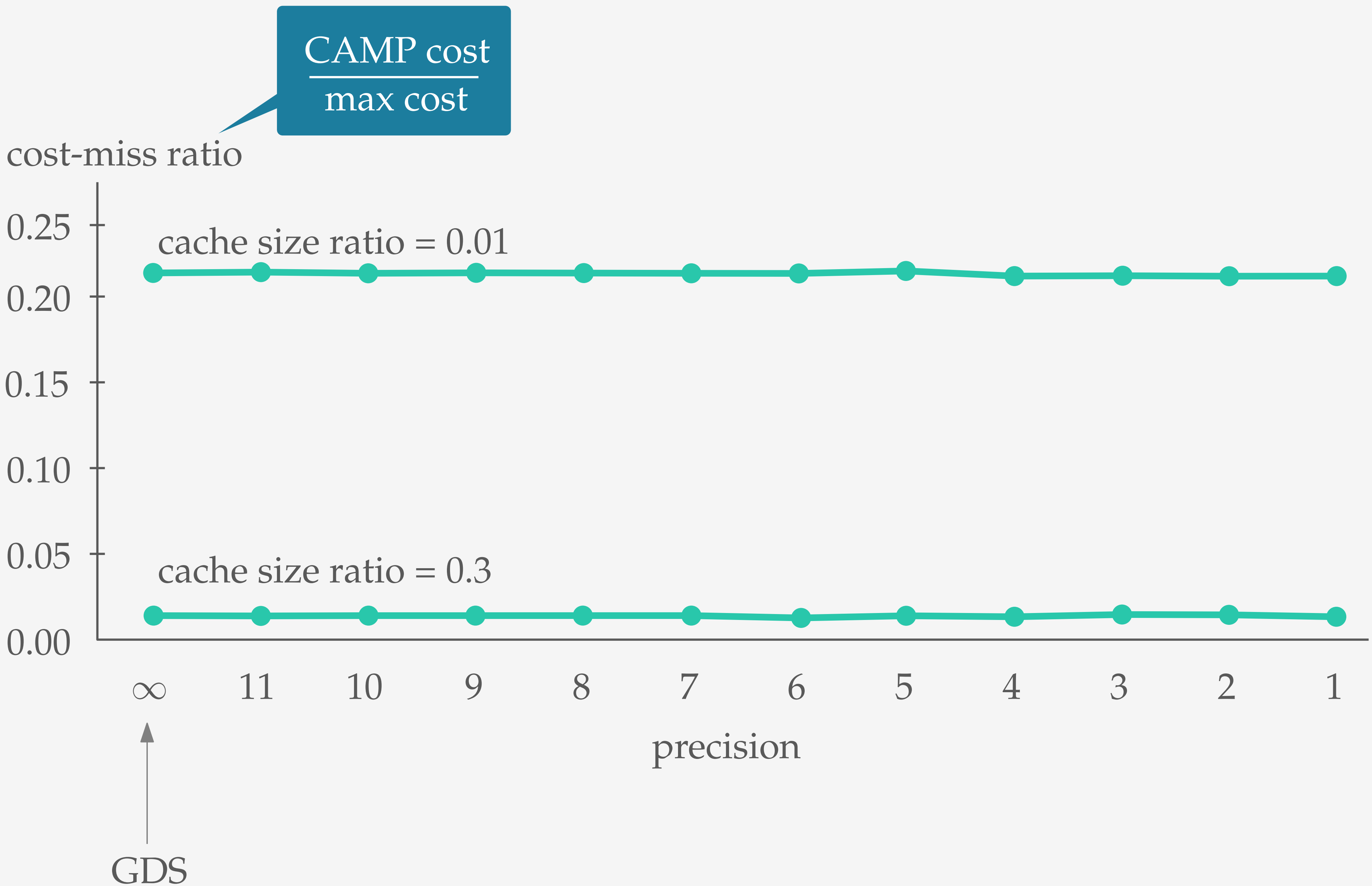
cost(GDS) $\leq$ k cost(OPT)

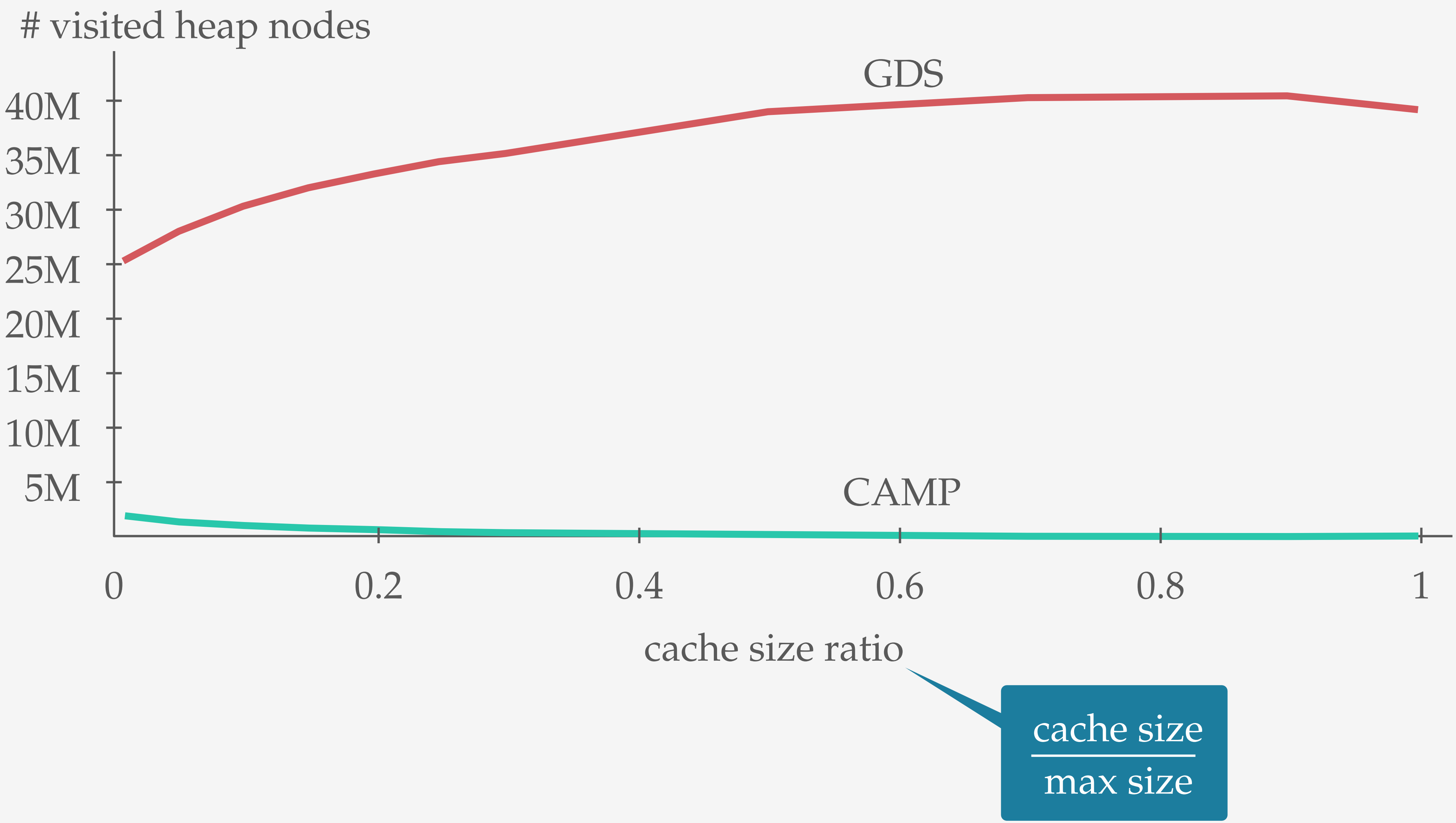
log (#queues) per update

cost(CAMP) $\leq (1 + \varepsilon)$k cost(OPT)

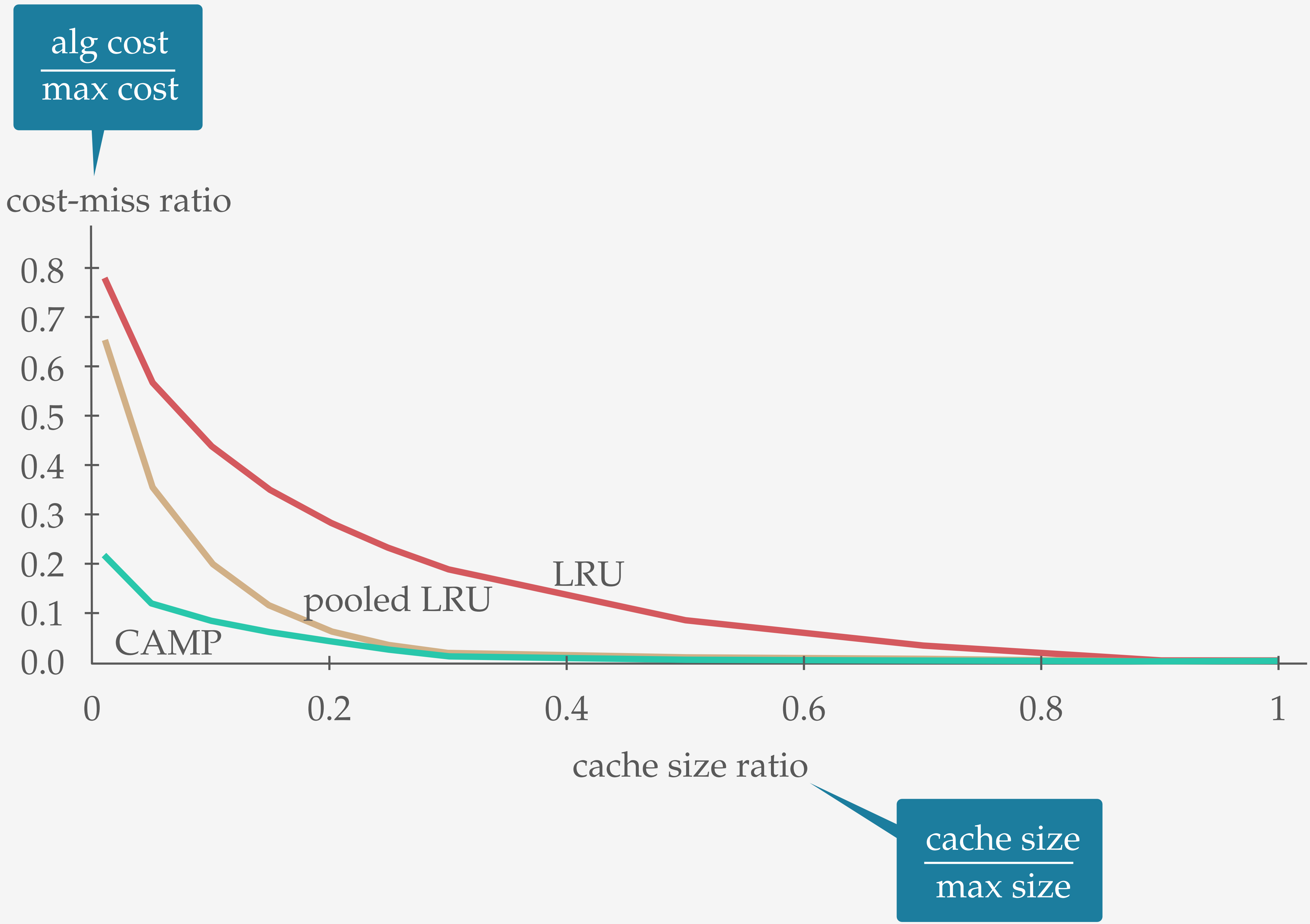
approximation parameter

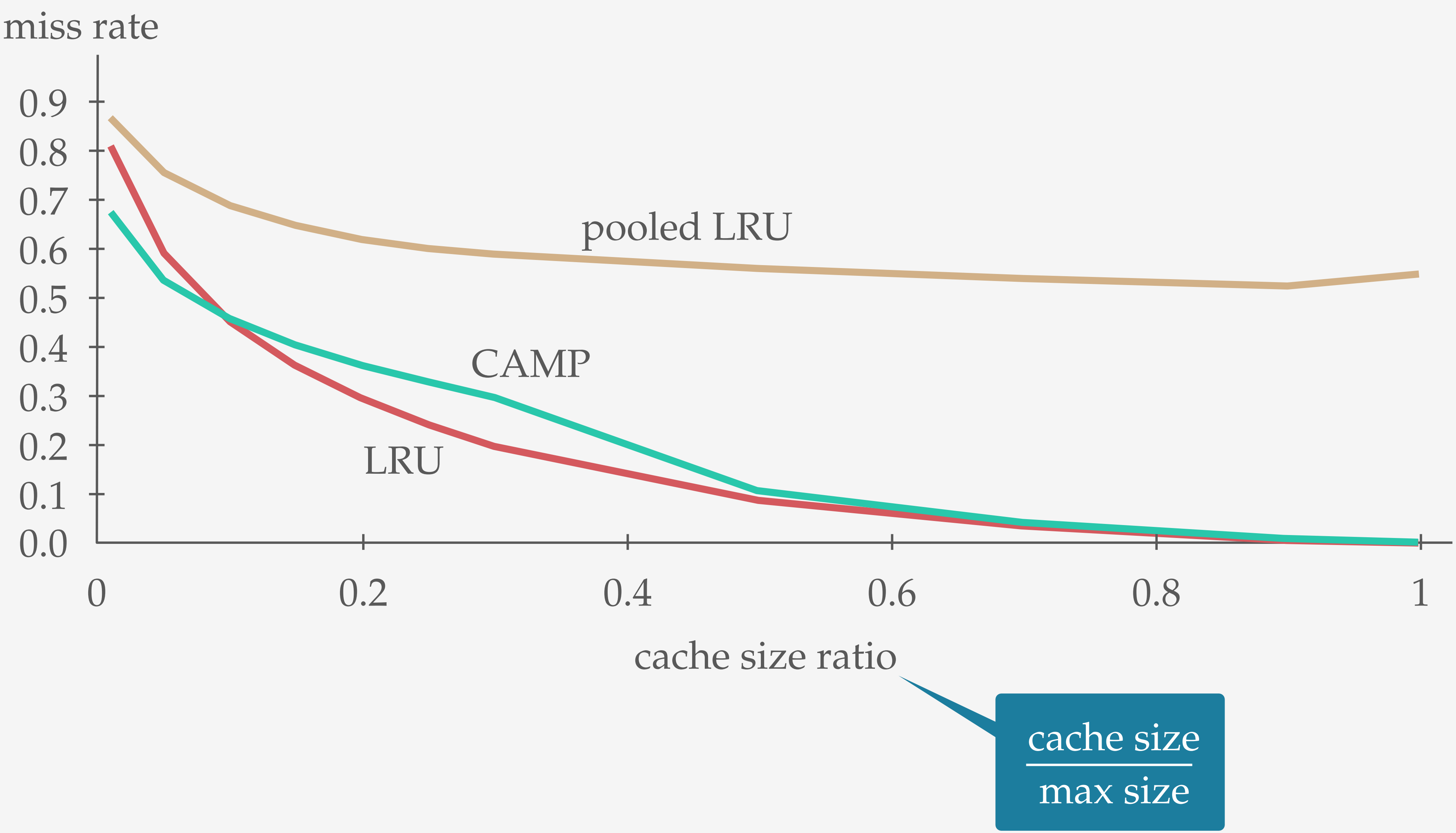

EXPERIMENTS
20,000 items
costs
sizes
key-value store
Twemcache
LRU
pooled LRU
CAMP
client
request generator
Whalin client for memcached
BG social networking benchmark
4 million requests
i.i.d. with 70% of requests to 20% of items


CAMP cost
max cost
cost-miss ratio
0.25
0.20
0.15
0.10
0.05
0.00
cache size ratio = 0.01
cache size ratio = 0.3
∞
11
10
9
8
7
6
5
4
3
2
1
precision
GDS

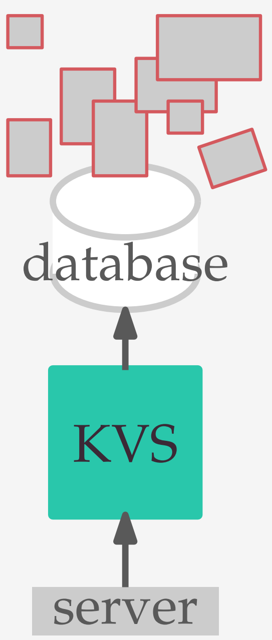


visited heap nodes
40M
35M
30M
25M
20M
15M
10M
5M
GDS
CAMP
0
0.2
0.4
0.6
0.8
1
cache size ratio
cache size
max size


alg cost
max cost
cost-miss ratio
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
LRU
pooled LRU
CAMP
0
0.2
0.4
0.6
0.8
1
cache size ratio
cache size
max size



miss rate
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
pooled LRU
CAMP
LRU
0
0.2
0.4
0.6
0.8
1
cache size ratio
cache size
max size



EVICTION POLICY

GDS → CAMP

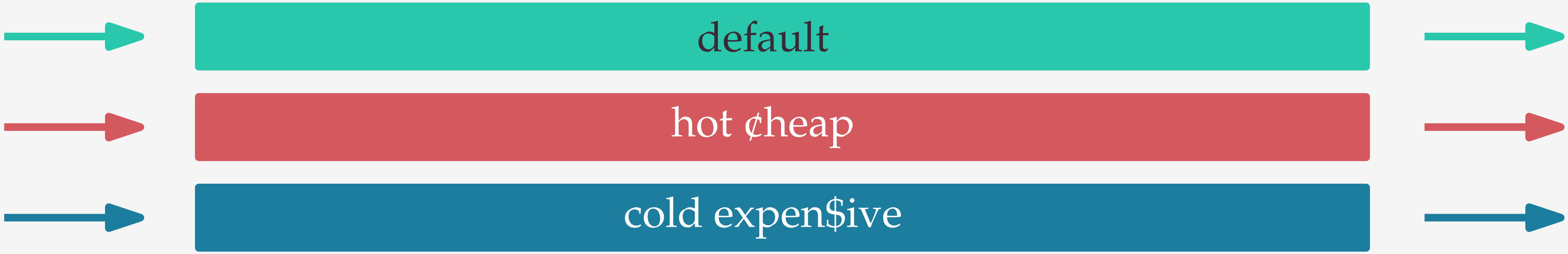
PLACEMENT POLICY

generalized caching → managed memory caching

MEMORY HIERARCHY

2-level cache → multi-level cache

pooled Least Recently Used



Scaling Memcache at Facebook, Nishtala et al., NSDI 2013.

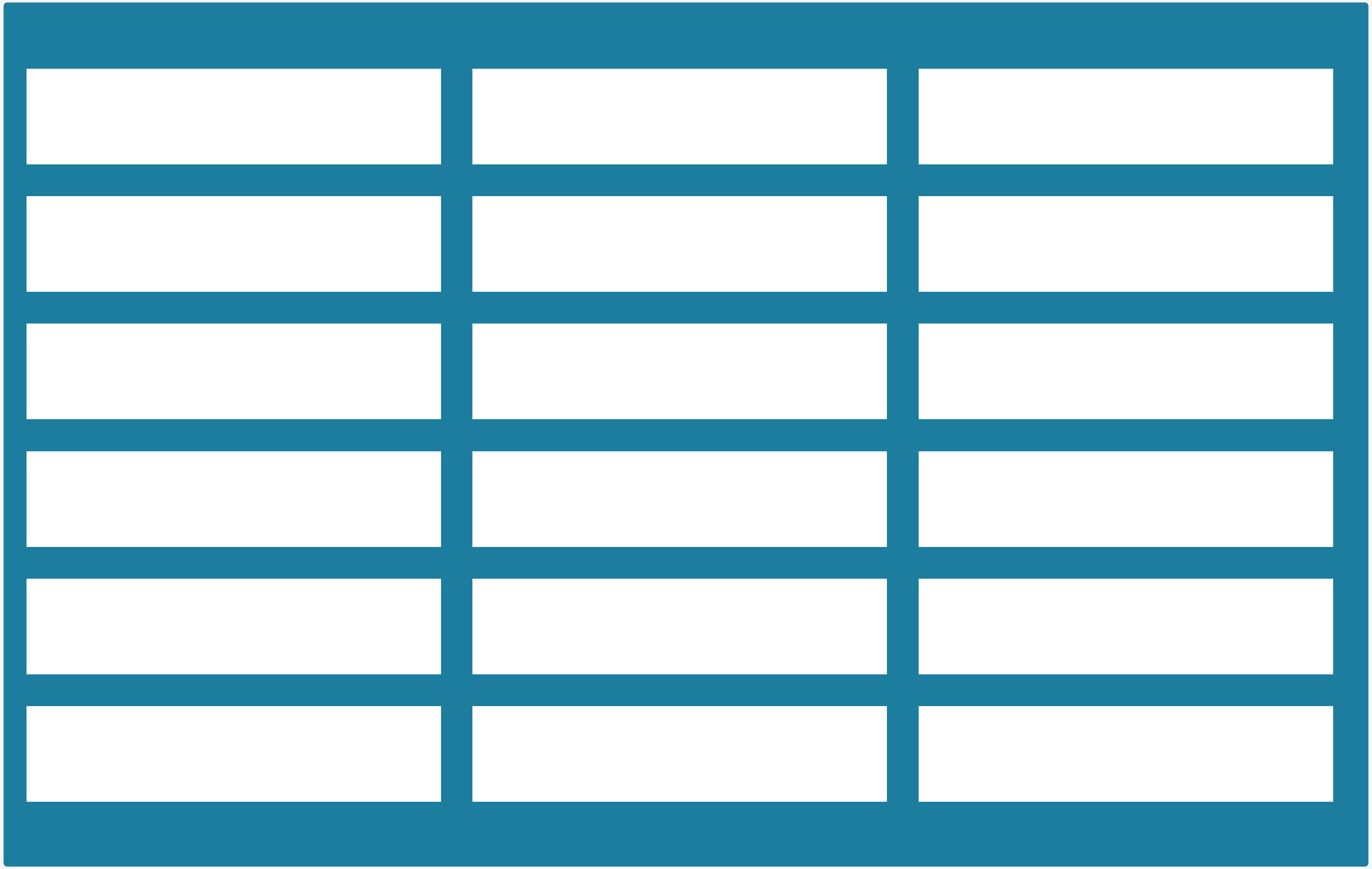
cold expen$ive

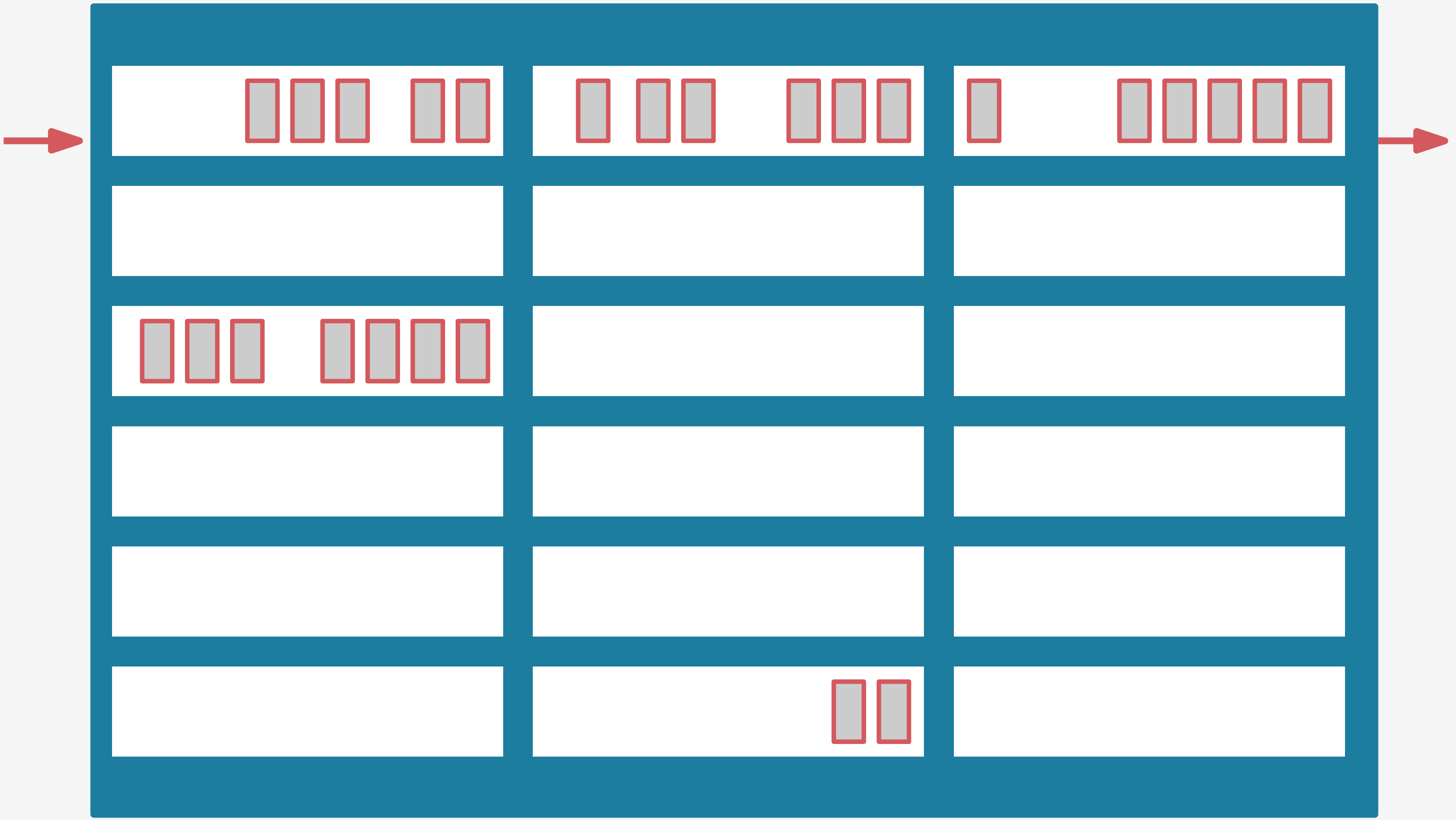

cold expen$ive

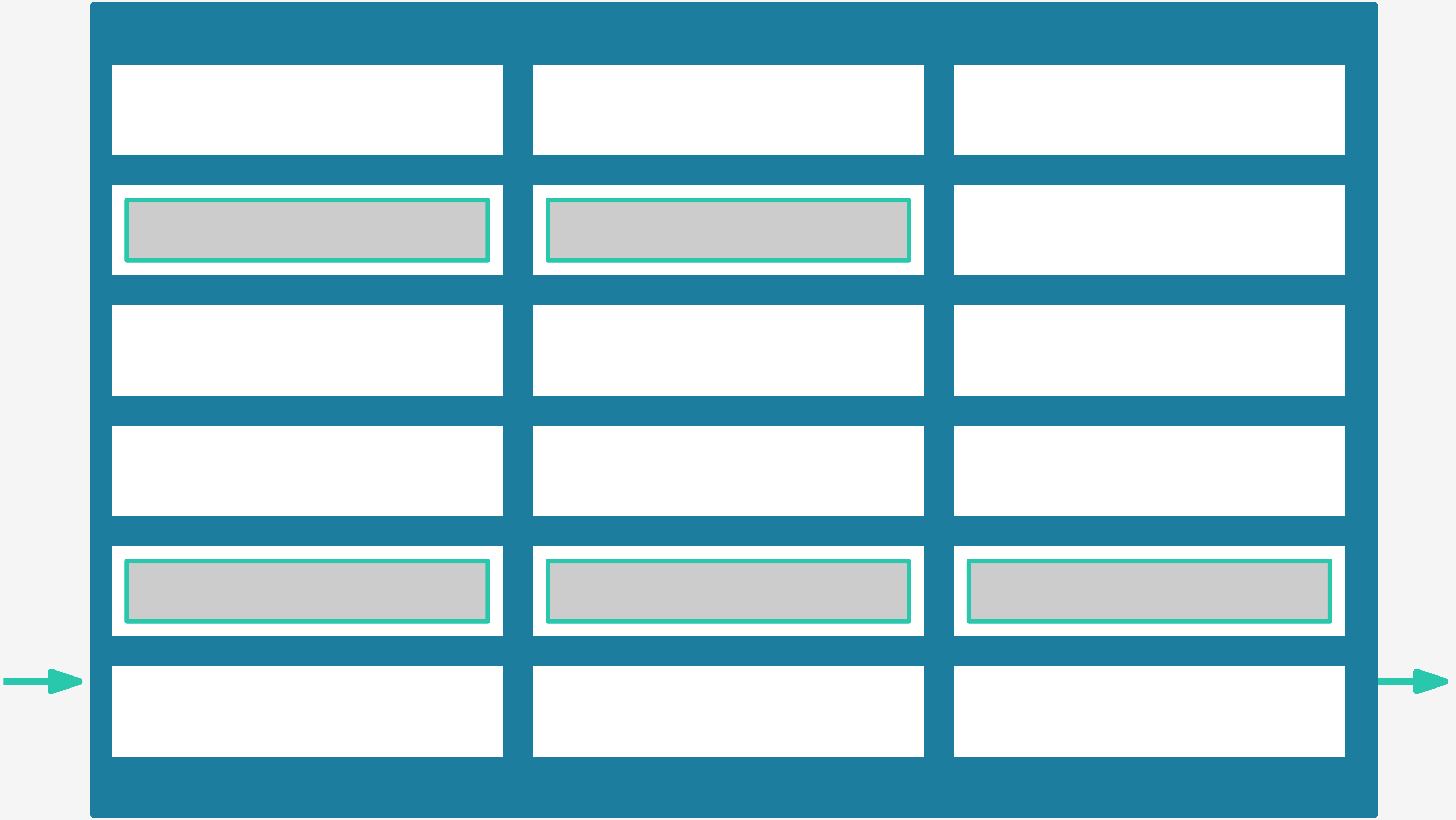

cold expen$ive

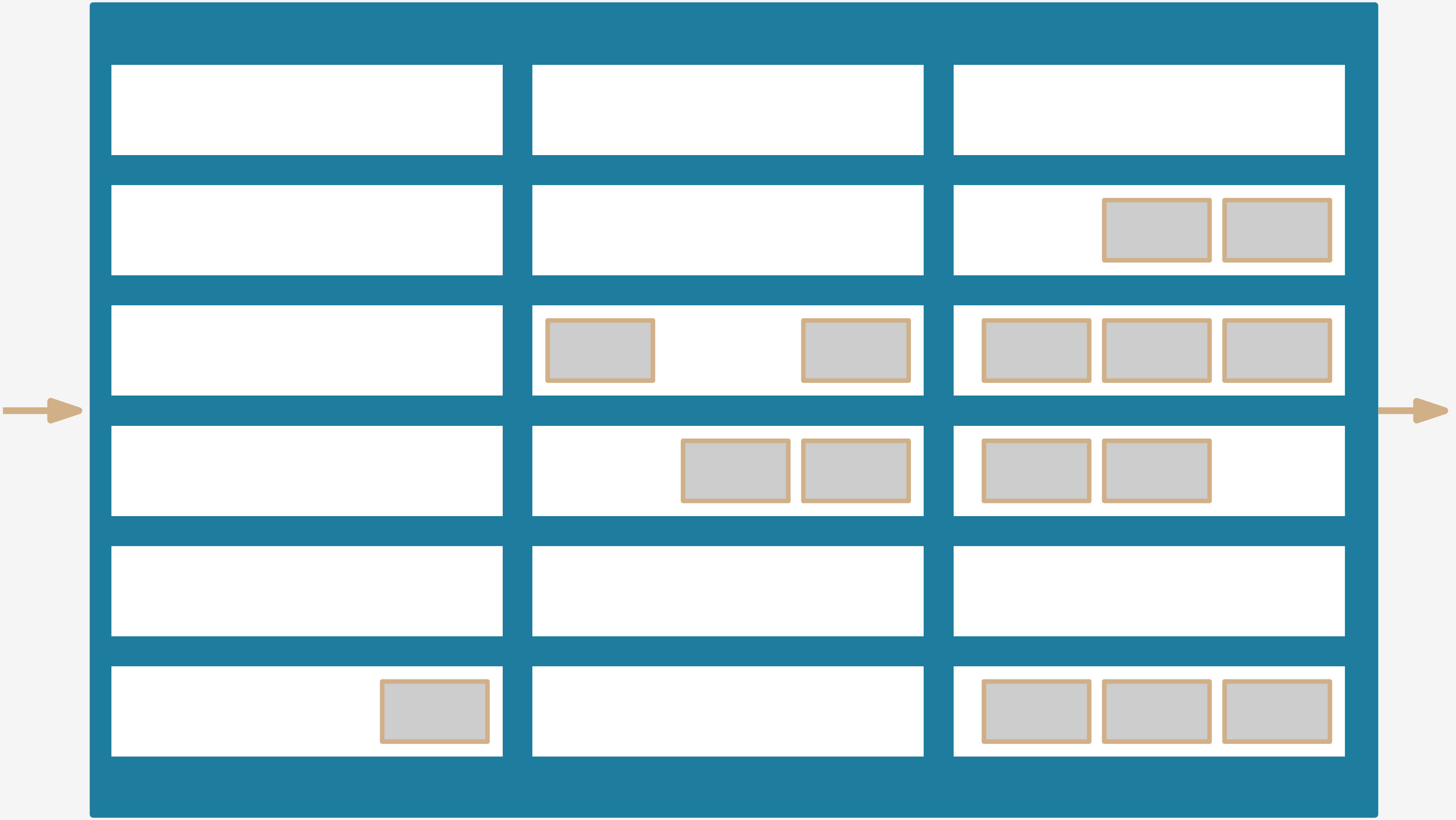

cold expen$ive

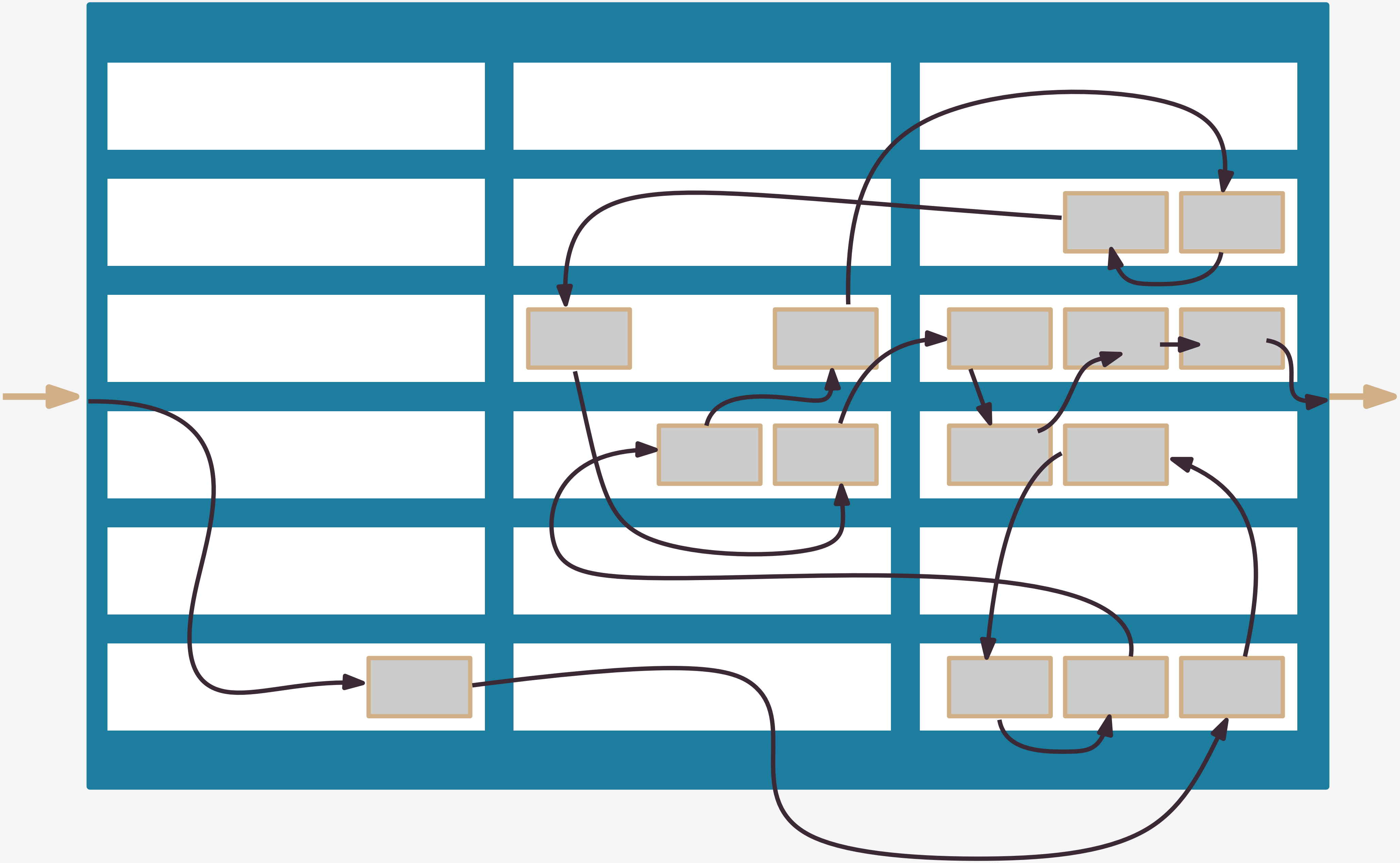
cold expen$ive

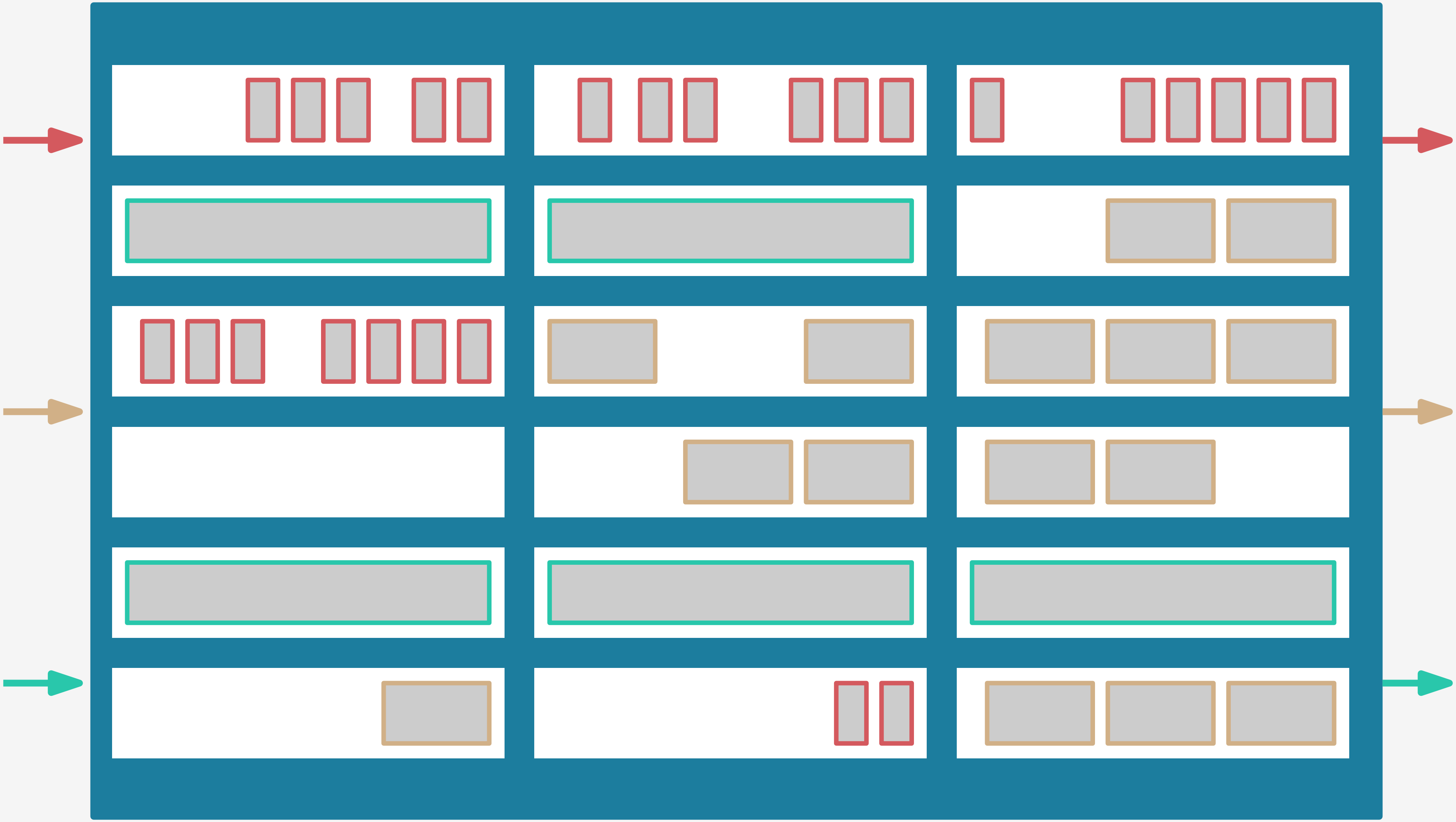


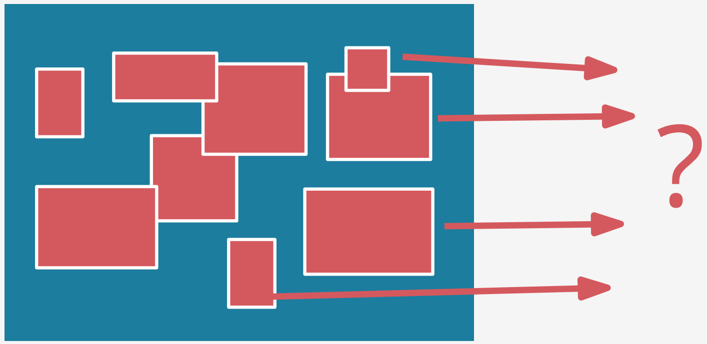




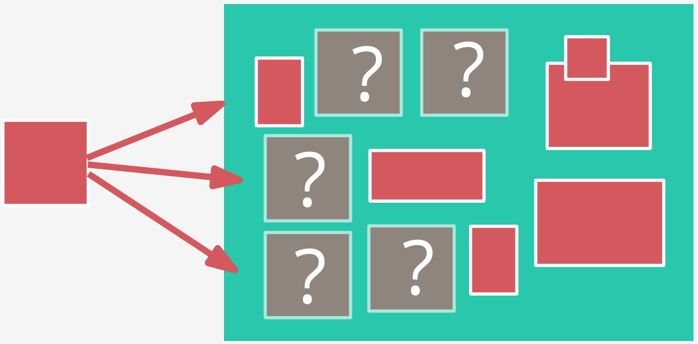








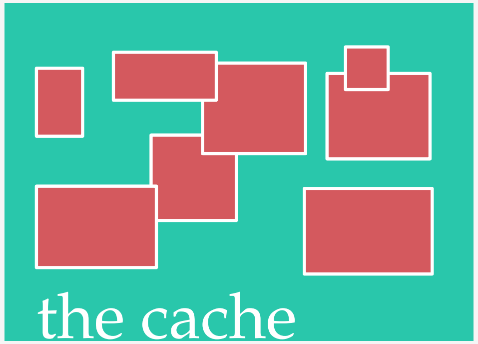
EVICTION POLICY



PLACEMENT POLICY

THE GENERALIZED CACHING PROBLEM

variable size and cost



GOAL

minimize **total cost** of cache misses

SUBJECT TO

total size of items in cache
cannot exceed the cache size

THE MANAGED MEMORY CACHING PROBLEM

variable size and cost

every item must fit in a contiguous segment of memory

CACHE REPLACEMENT

MEMORY ALLOCATION

CAMP-MALLOC

LRU queues

CAMP-MALLOC

FIFO queues

CAMP-MALLOC

FIFO queue

CAMP-MALLOC

FIFO queue

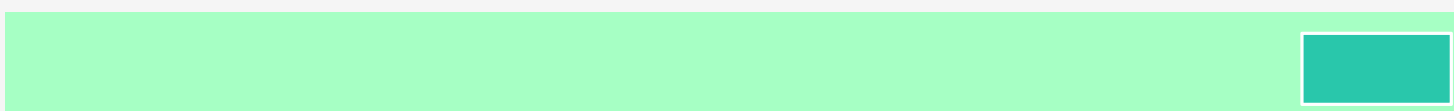
CAMP-MALLOC

FIFO queue

CAMP-MALLOC

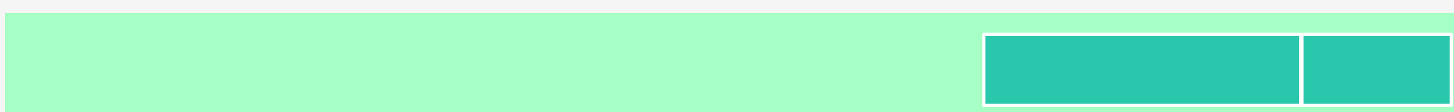
FIFO queue

CAMP-MALLOC



FIFO queue

CAMP-MALLOC

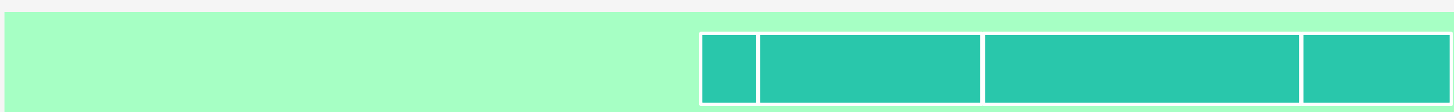


FIFO queue

CAMP-MALLOC

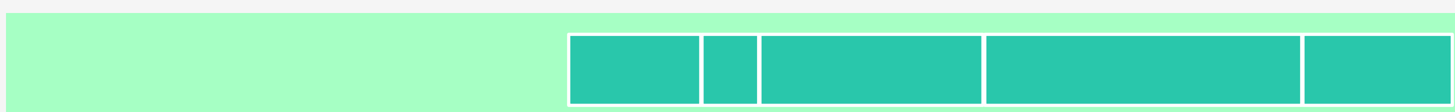
FIFO queue

CAMP-MALLOC



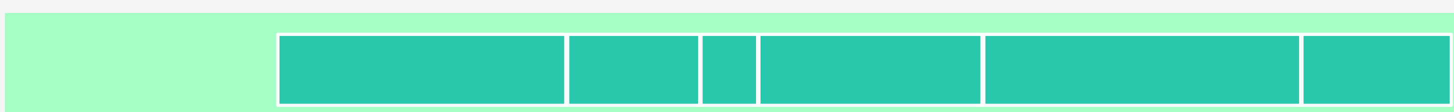
FIFO queue

CAMP-MALLOC



FIFO queue

CAMP-MALLOC

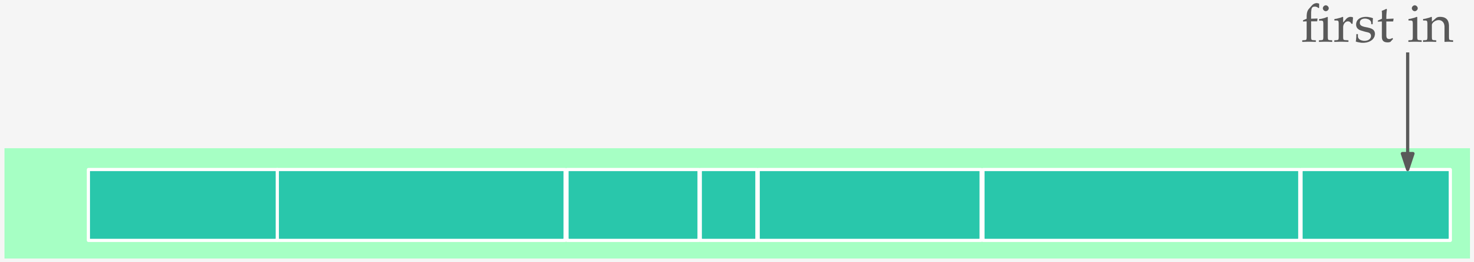


FIFO queue

CAMP-MALLOC

FIFO queue

CAMP-MALLOC

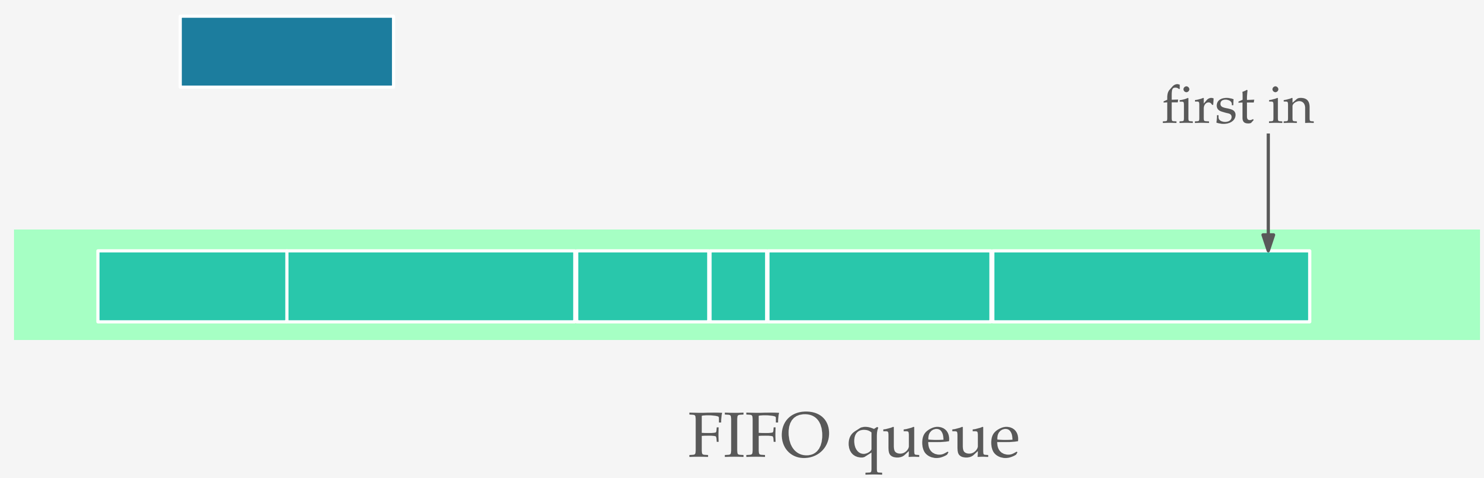


FIFO queue

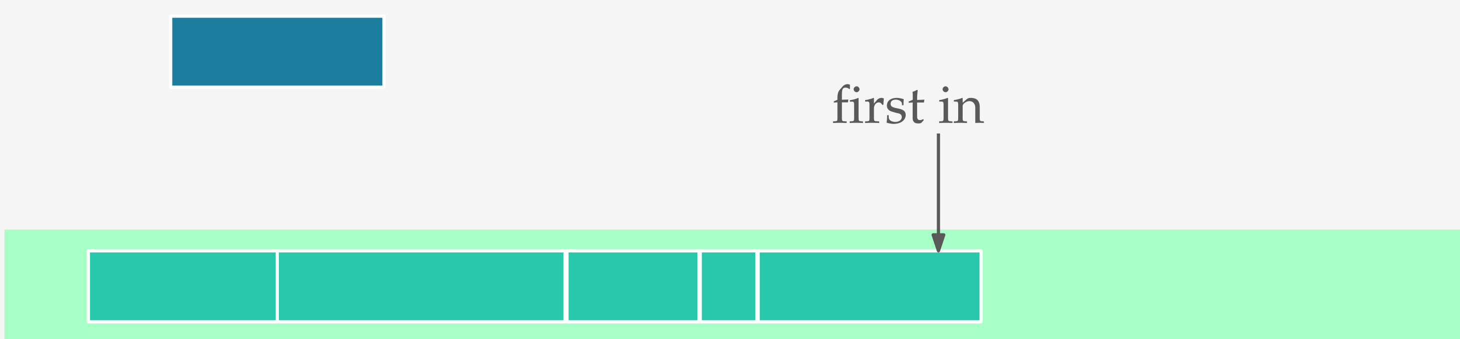
CAMP-MALLOC

FIFO queue

CAMP-MALLOC

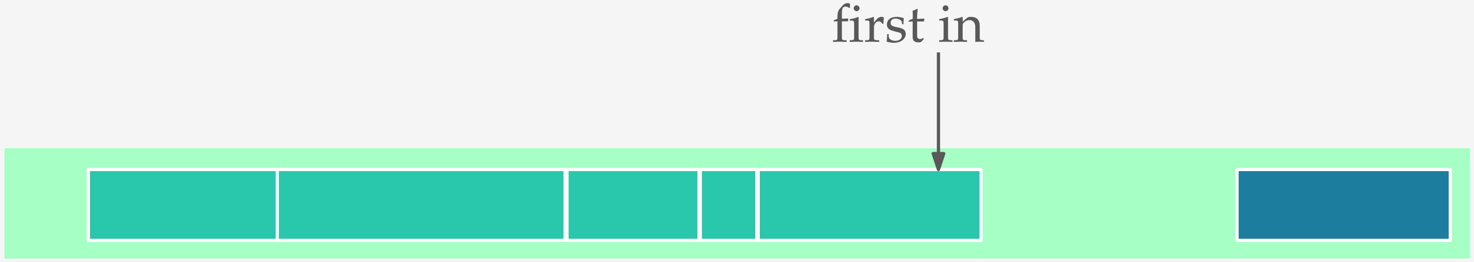


CAMP-MALLOC



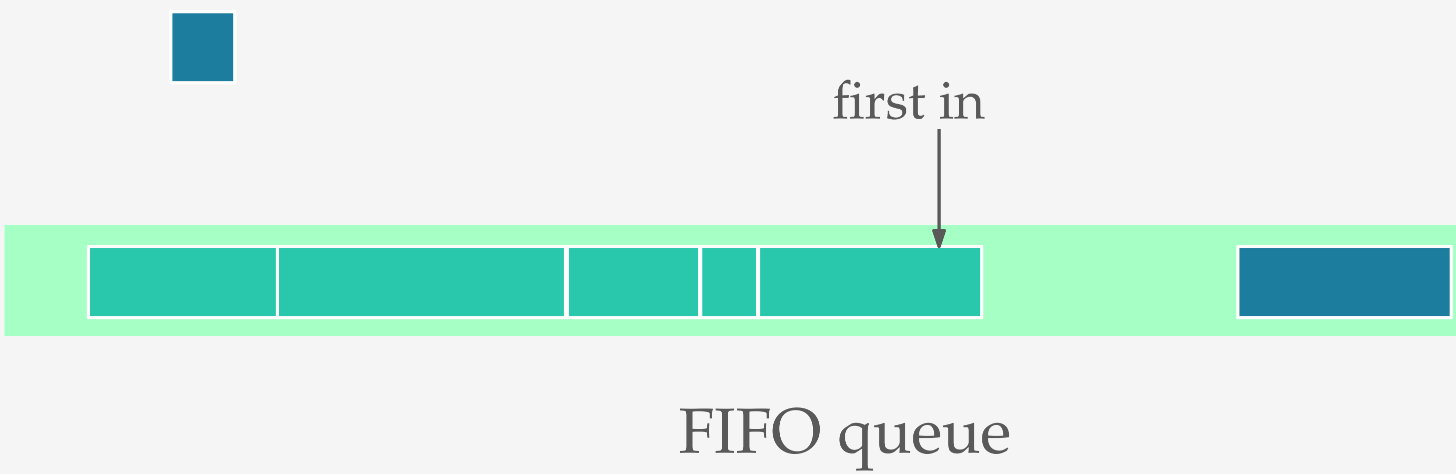
FIFO queue

CAMP-MALLOC

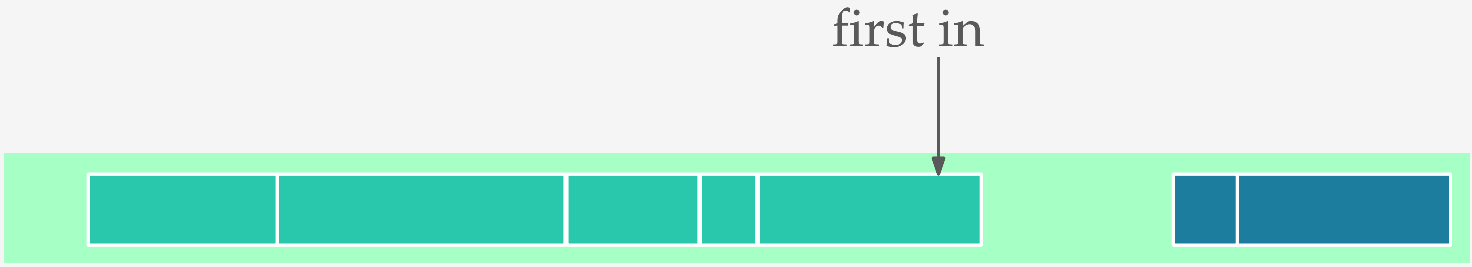


FIFO queue

CAMP-MALLOC

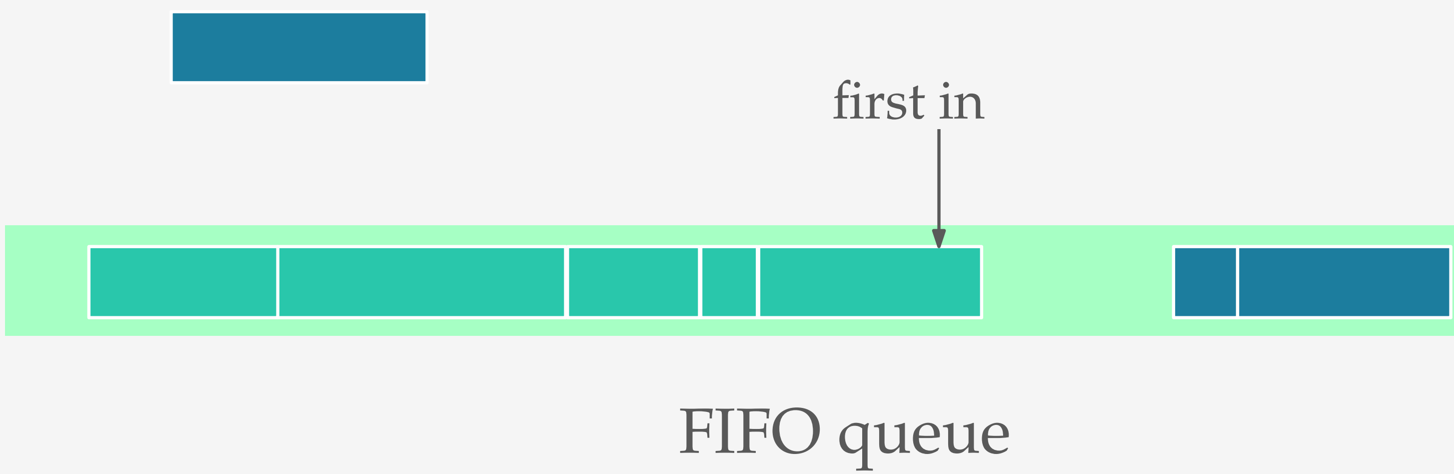


CAMP-MALLOC

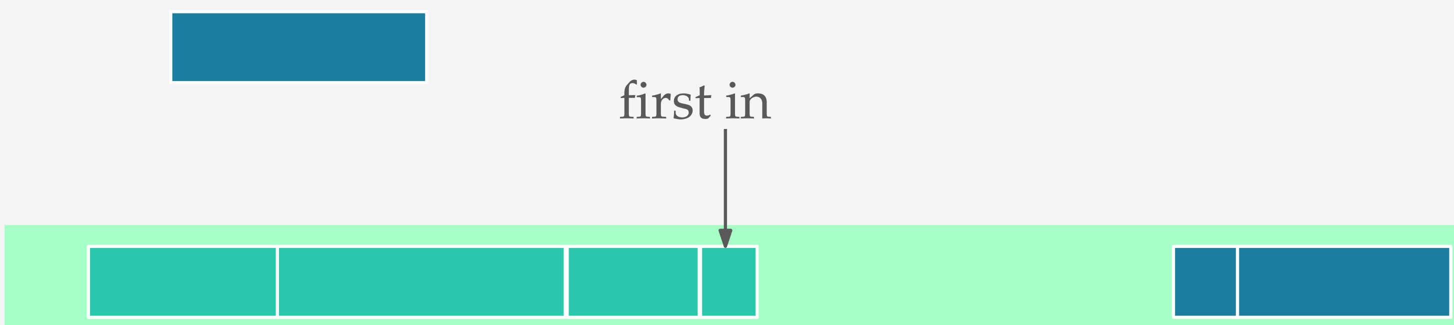


FIFO queue

CAMP-MALLOC

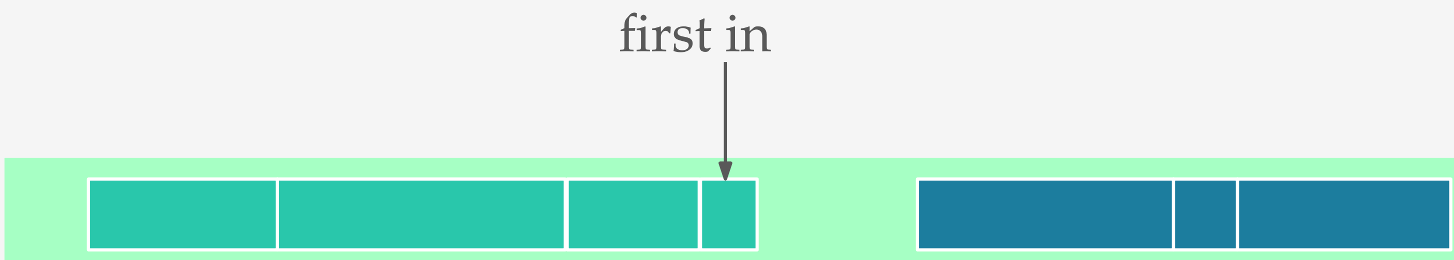


CAMP-MALLOC



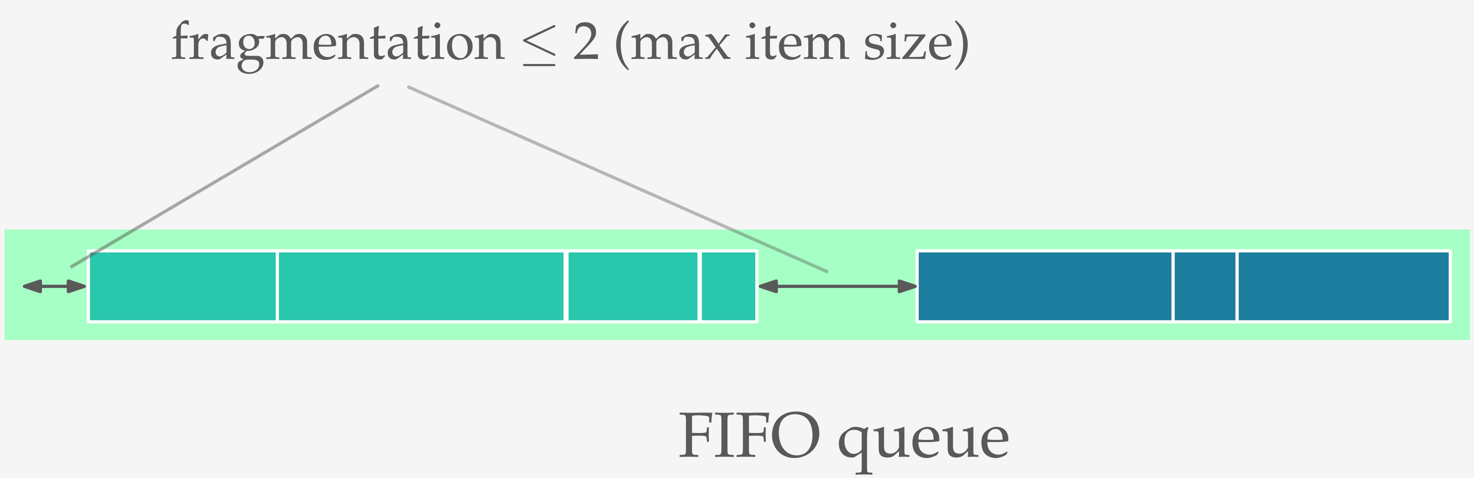
FIFO queue

CAMP-MALLOC



FIFO queue

CAMP-MALLOC



CAMP-MALLOC

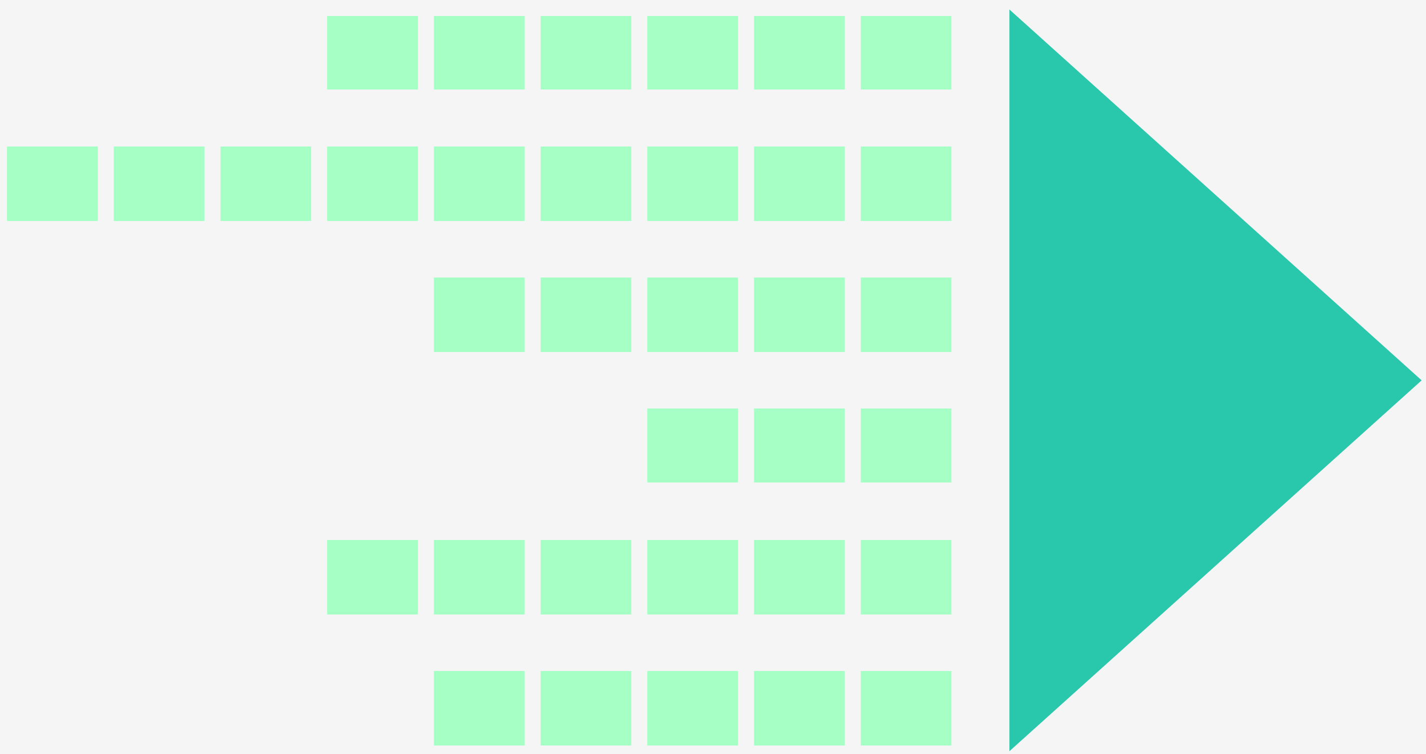
CAMP-MALLOC

CAMP-MALLOC

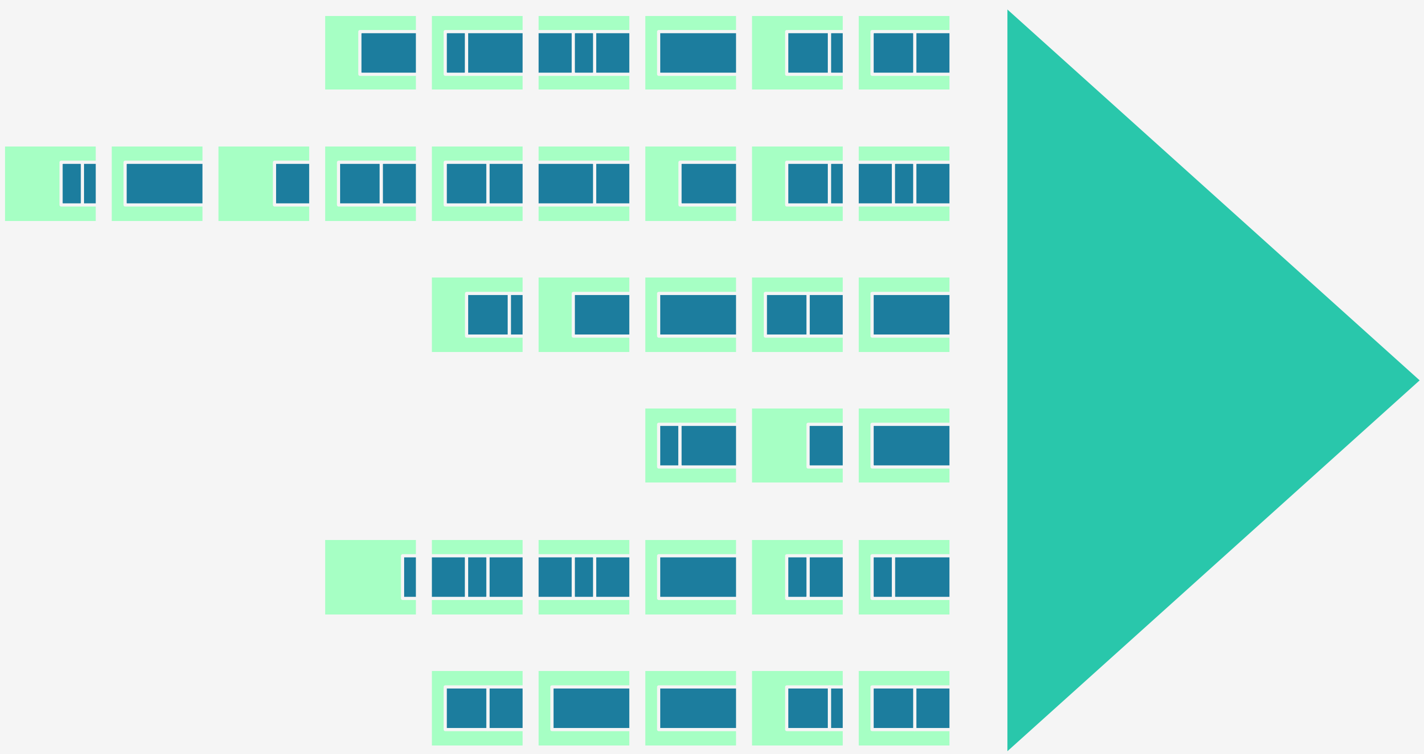
CAMP-MALLOC

CAMP-MALLOC

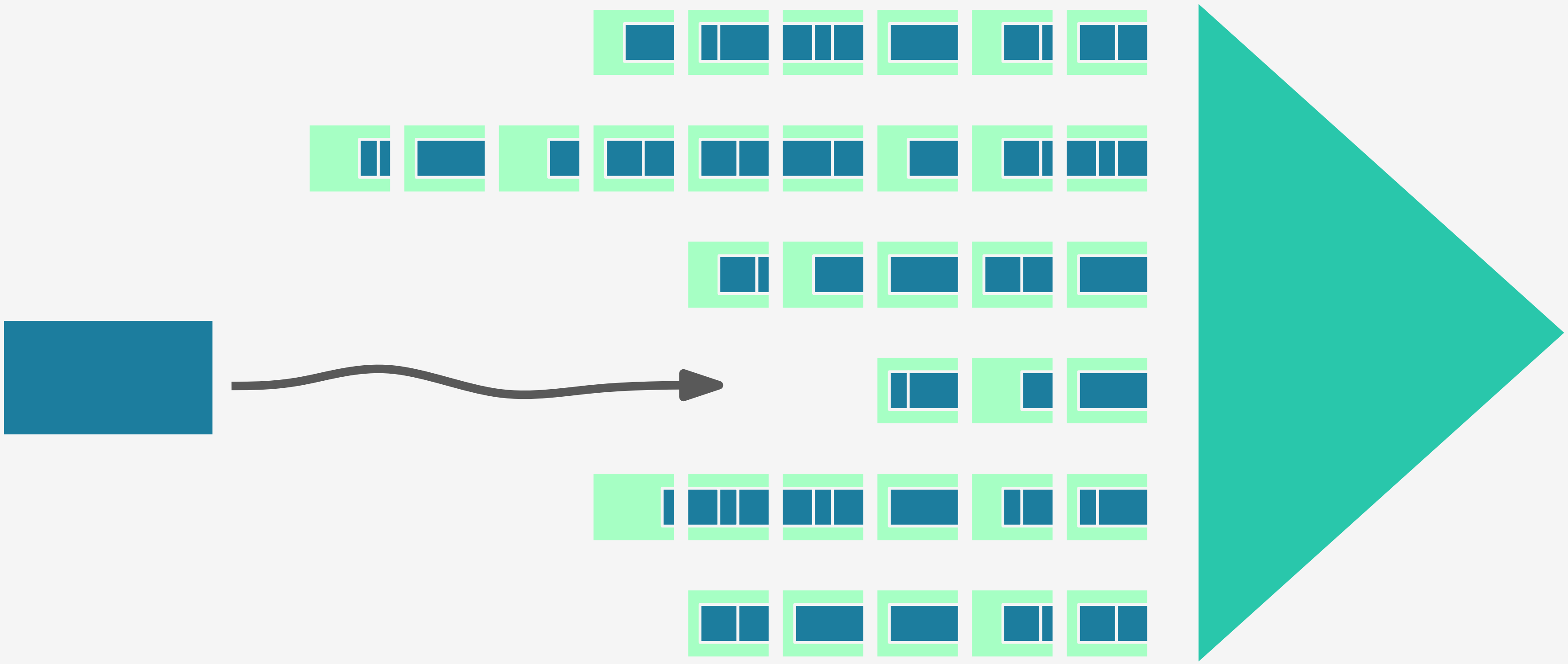
CAMP-MALLOC



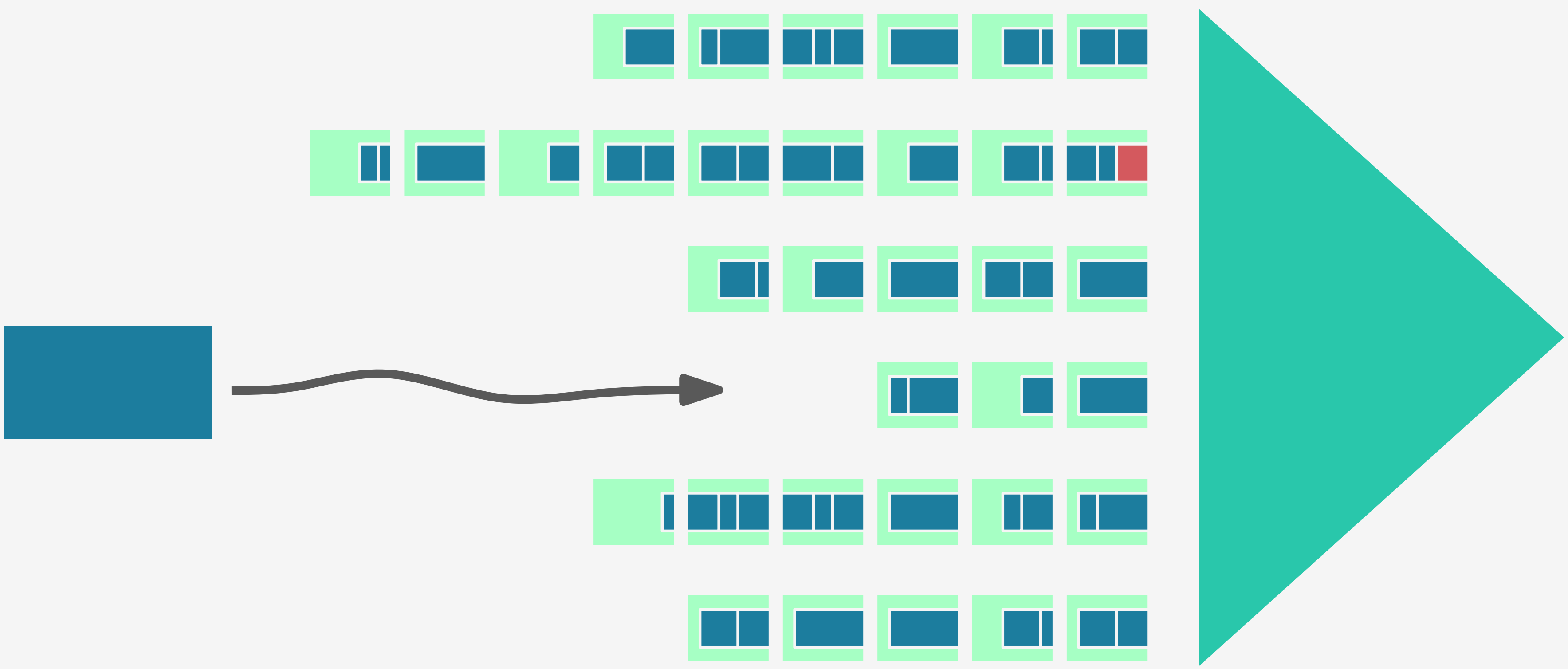
CAMP-MALLOC



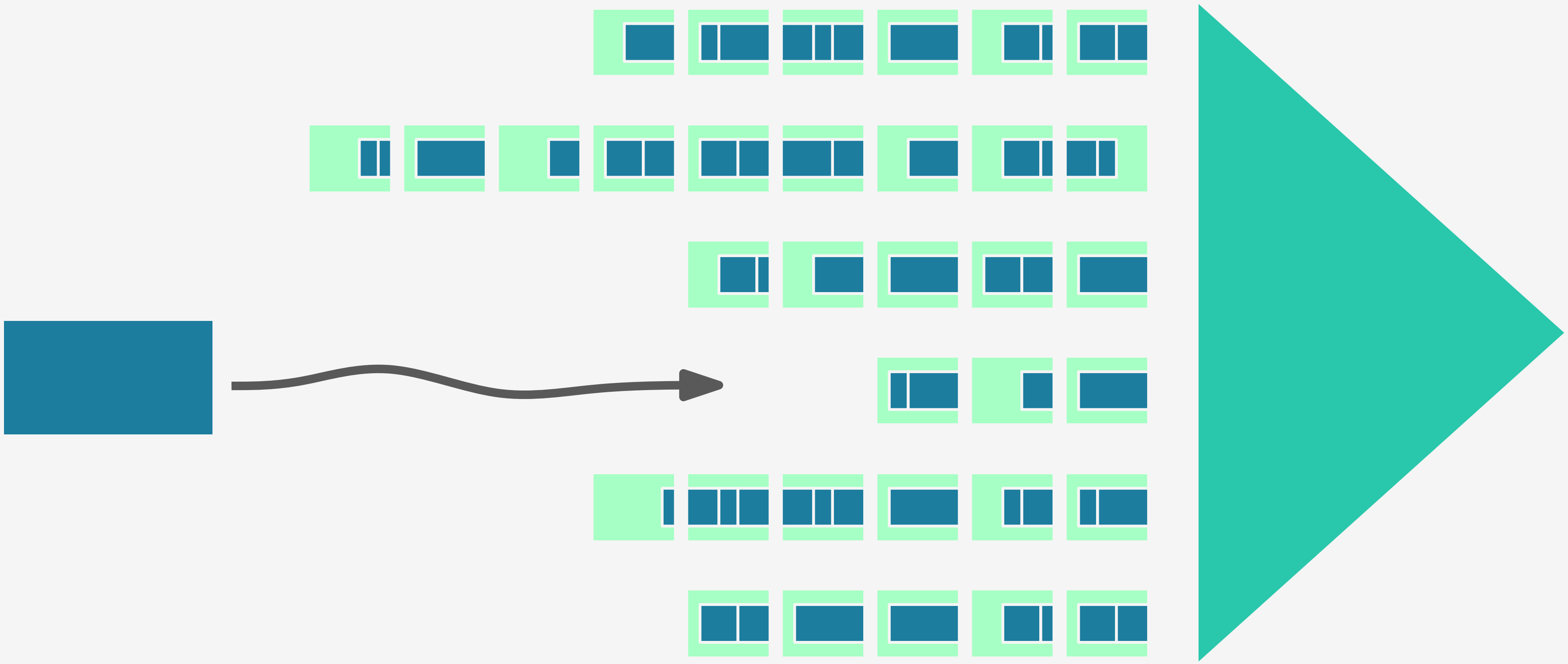
CAMP-MALLOC



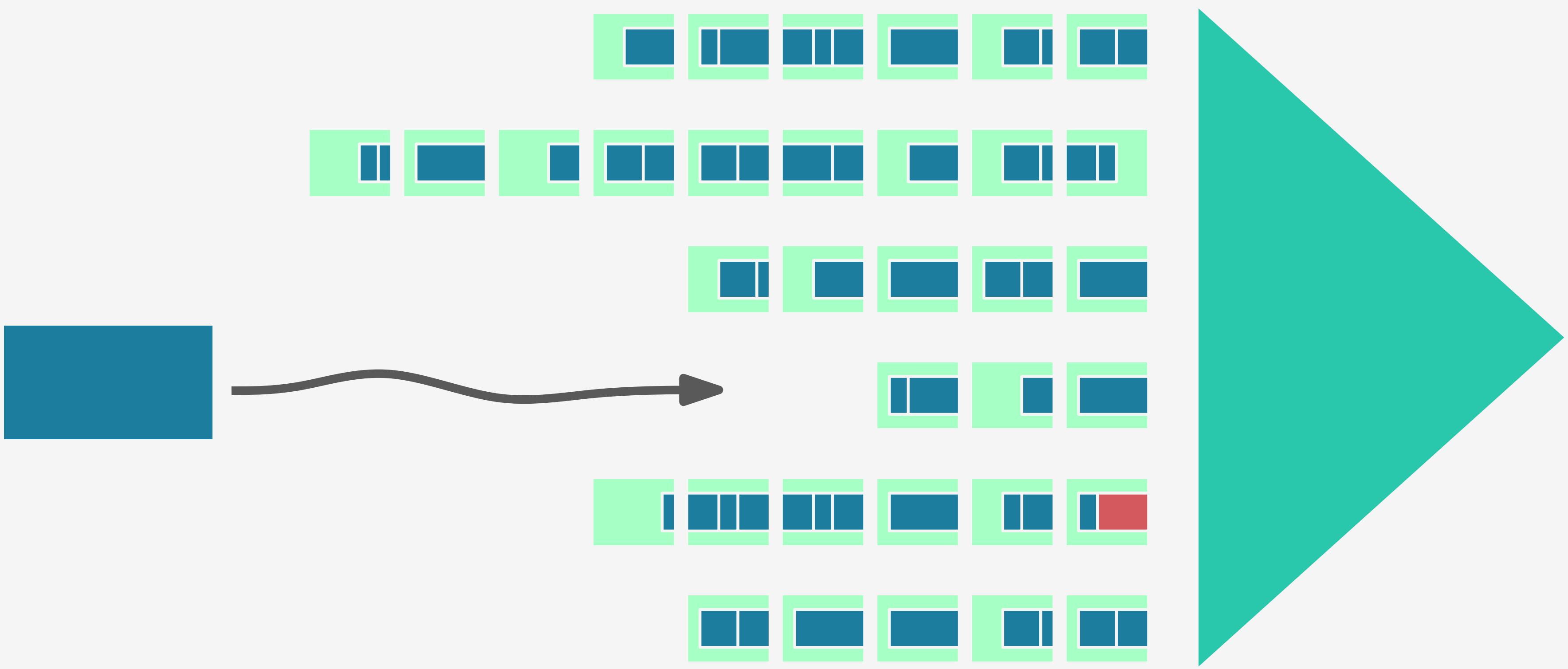
CAMP-MALLOC



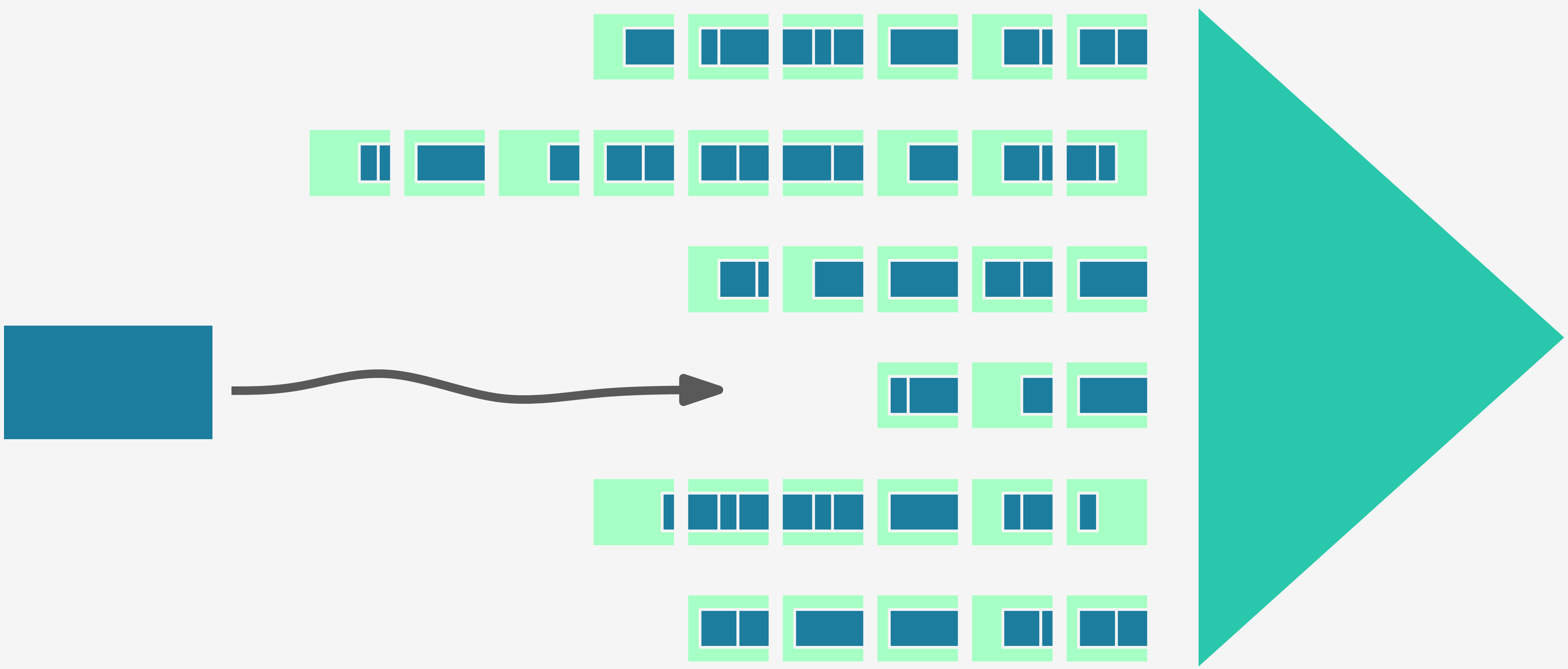
CAMP-MALLOC



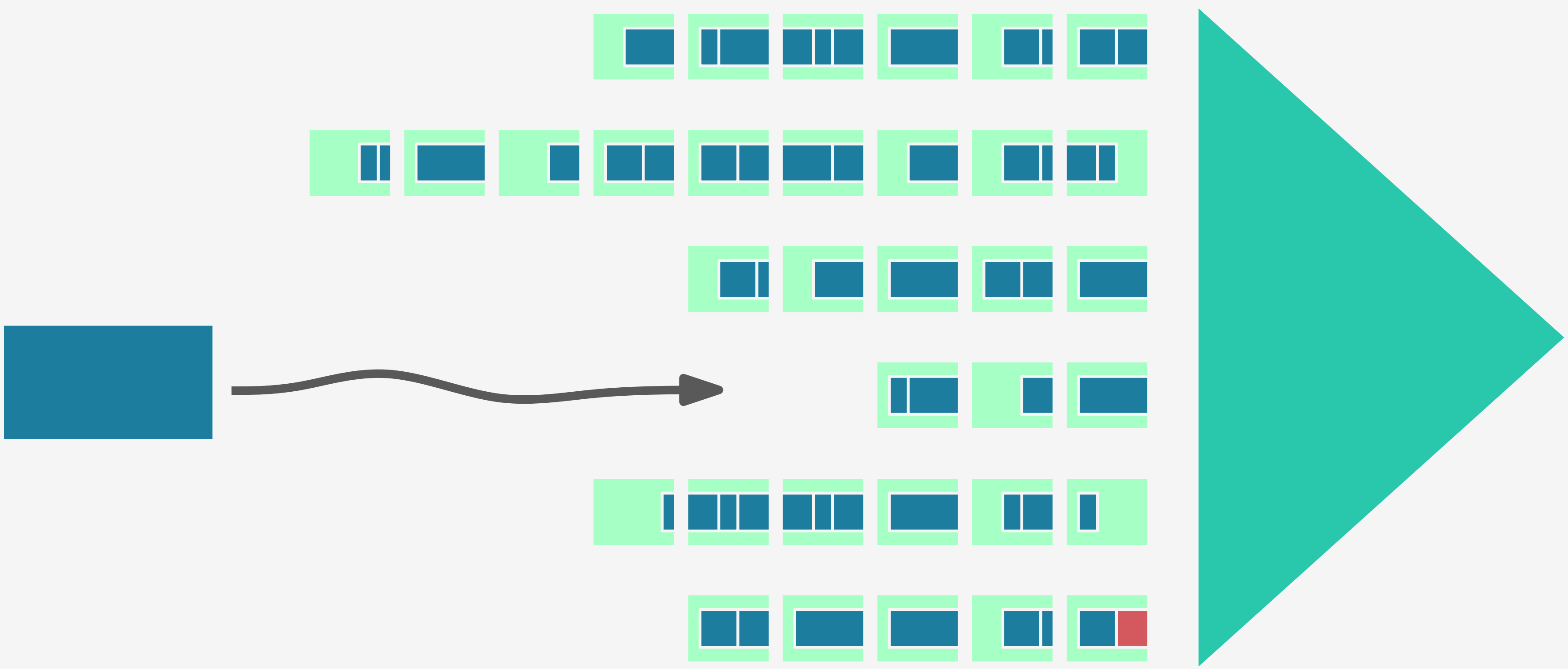
CAMP-MALLOC



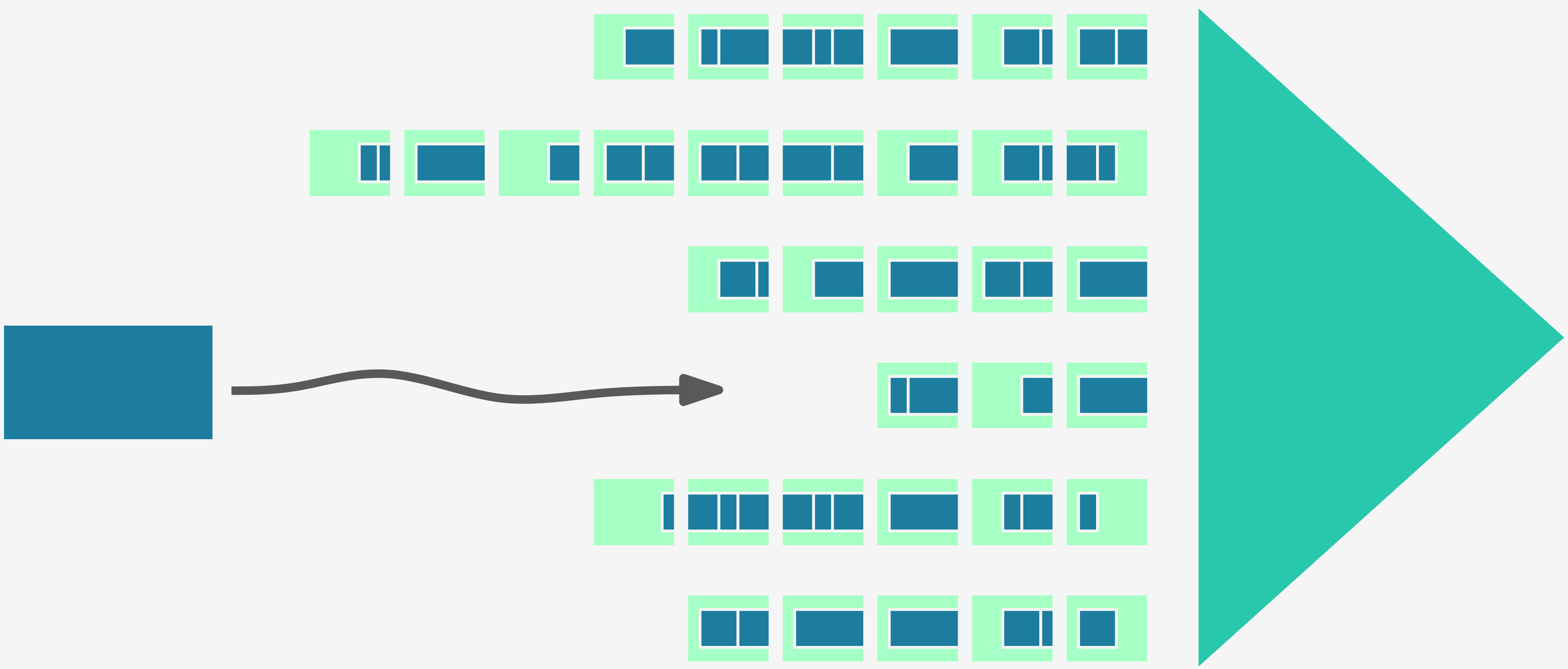
CAMP-MALLOC



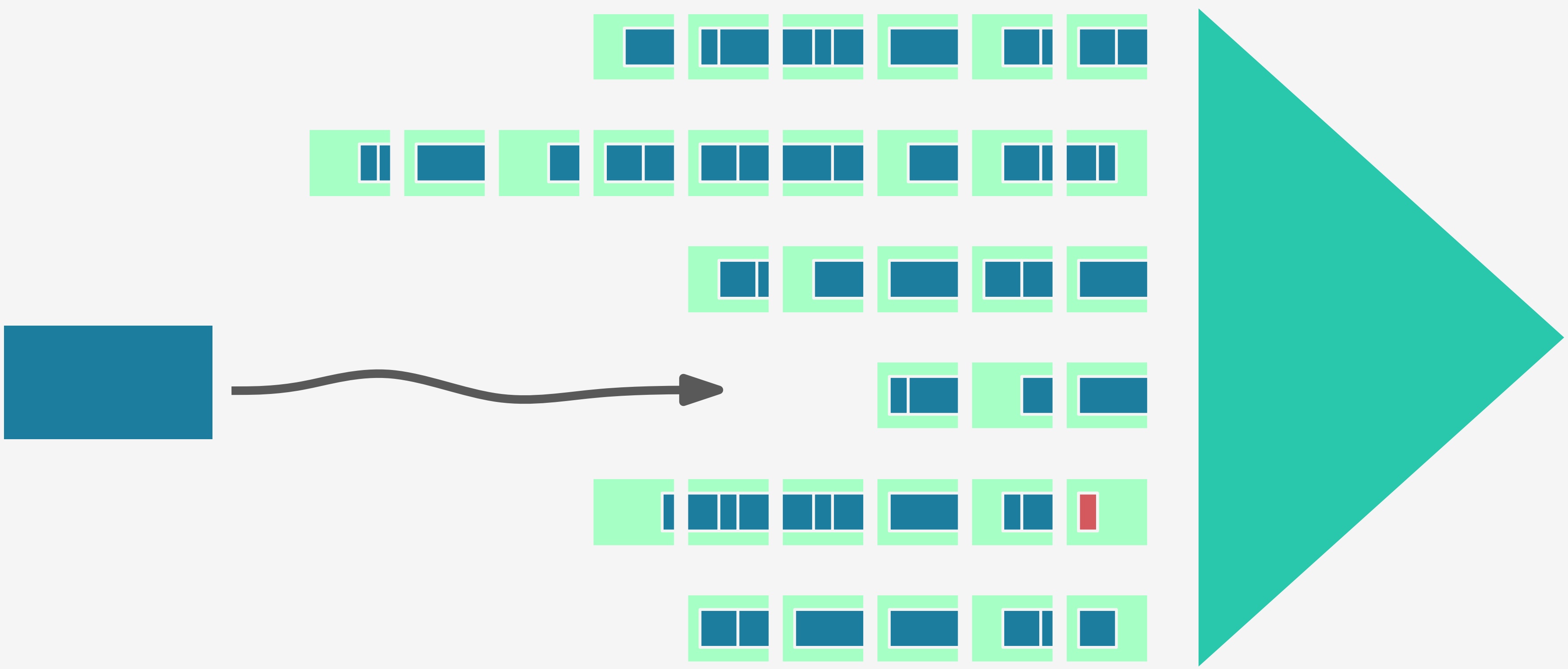
CAMP-MALLOC



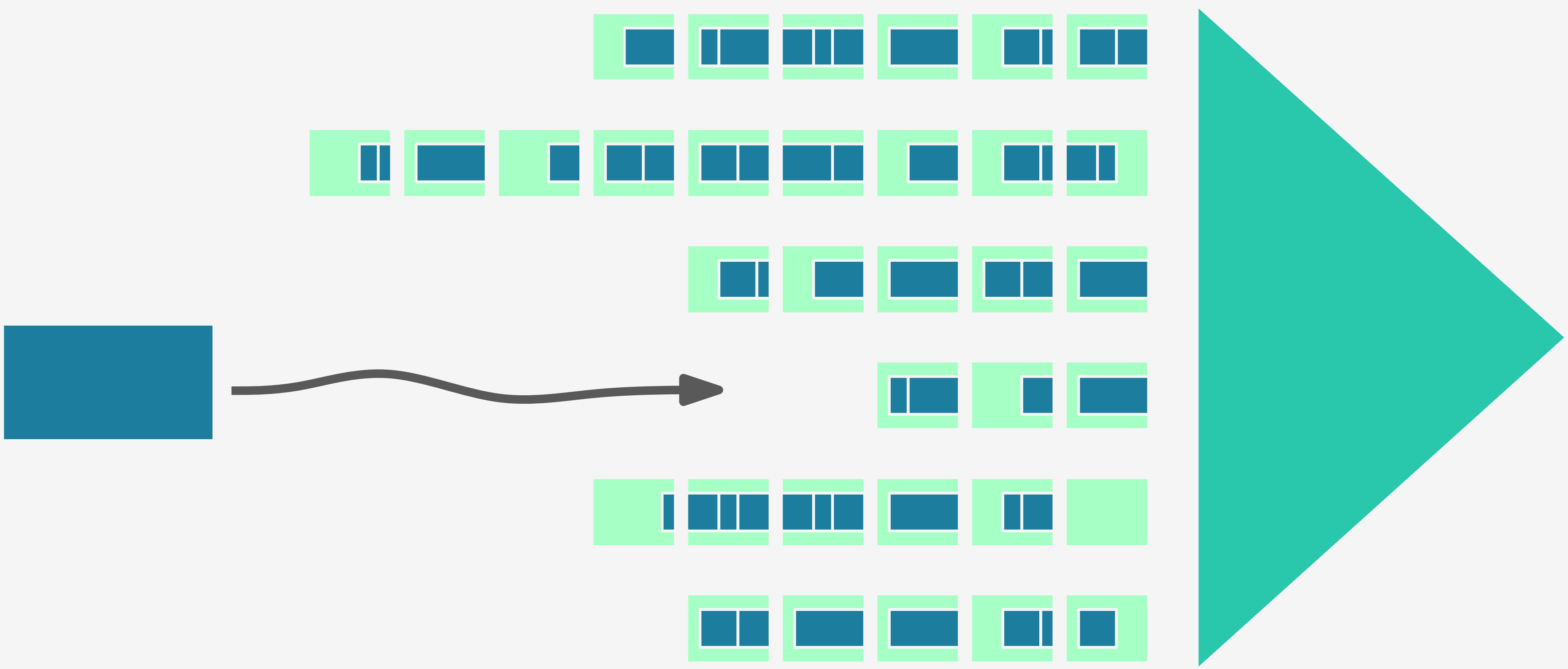
CAMP-MALLOC



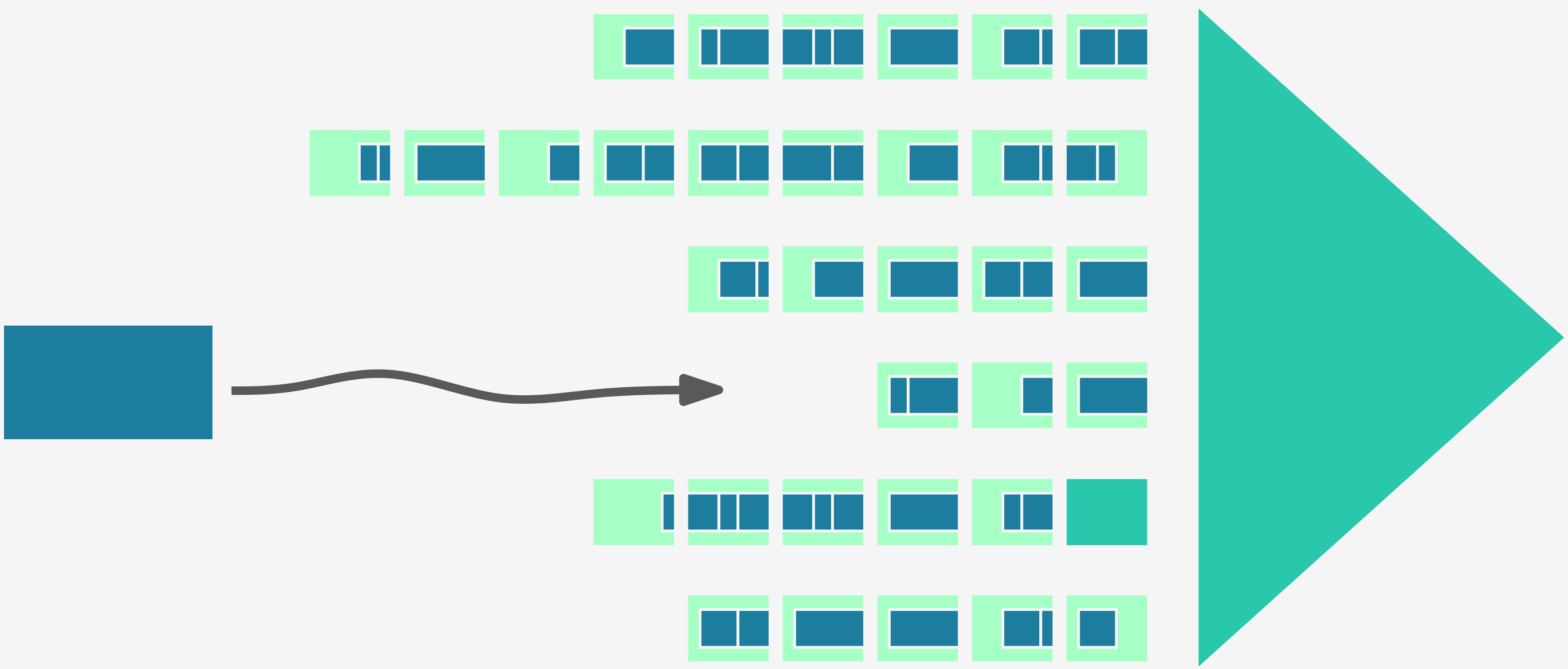
CAMP-MALLOC



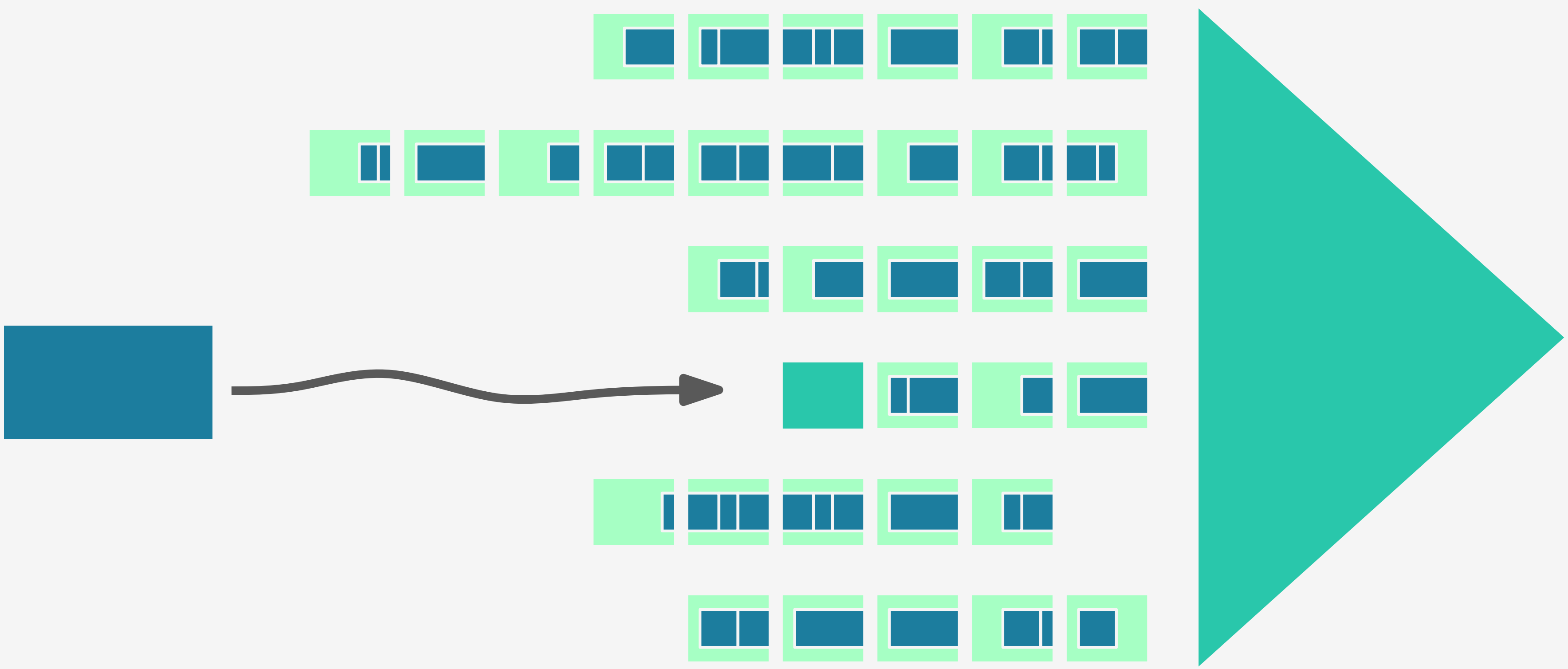
CAMP-MALLOC



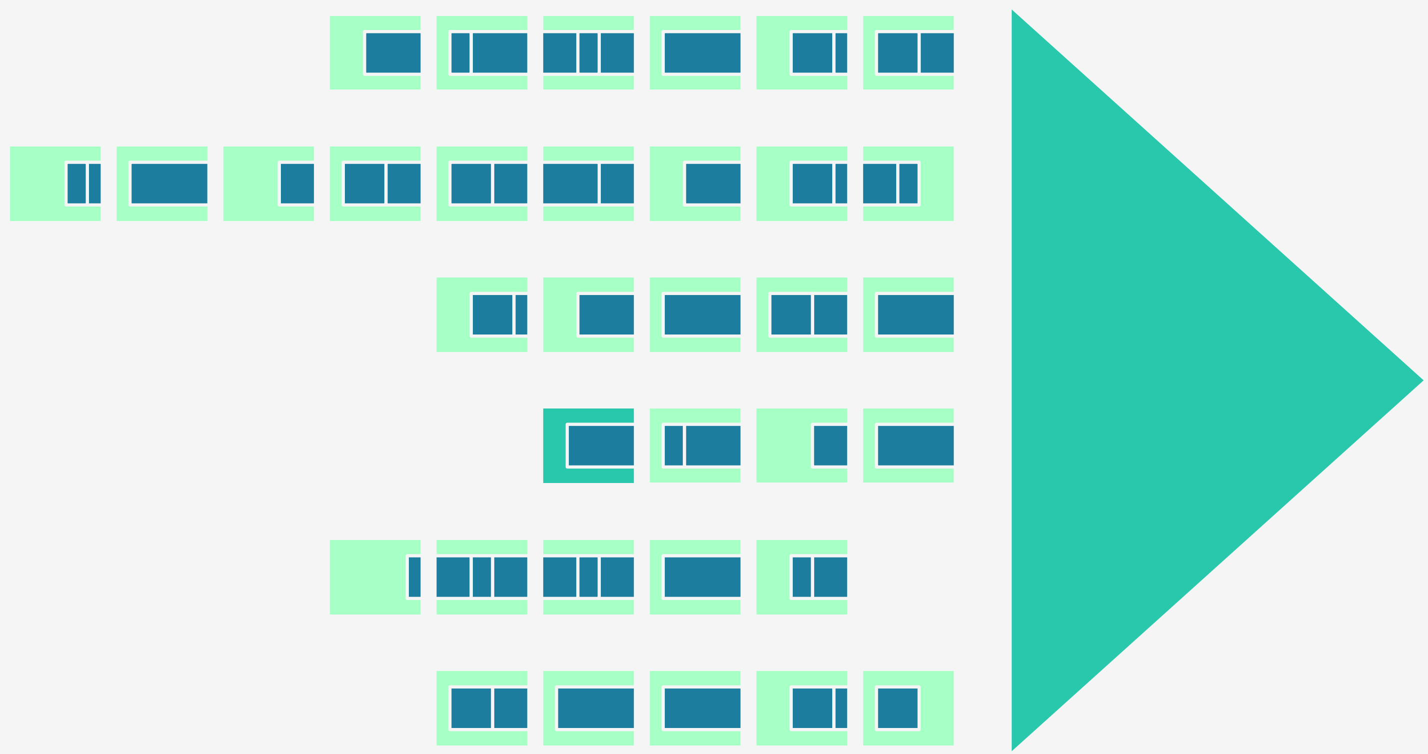
CAMP-MALLOC



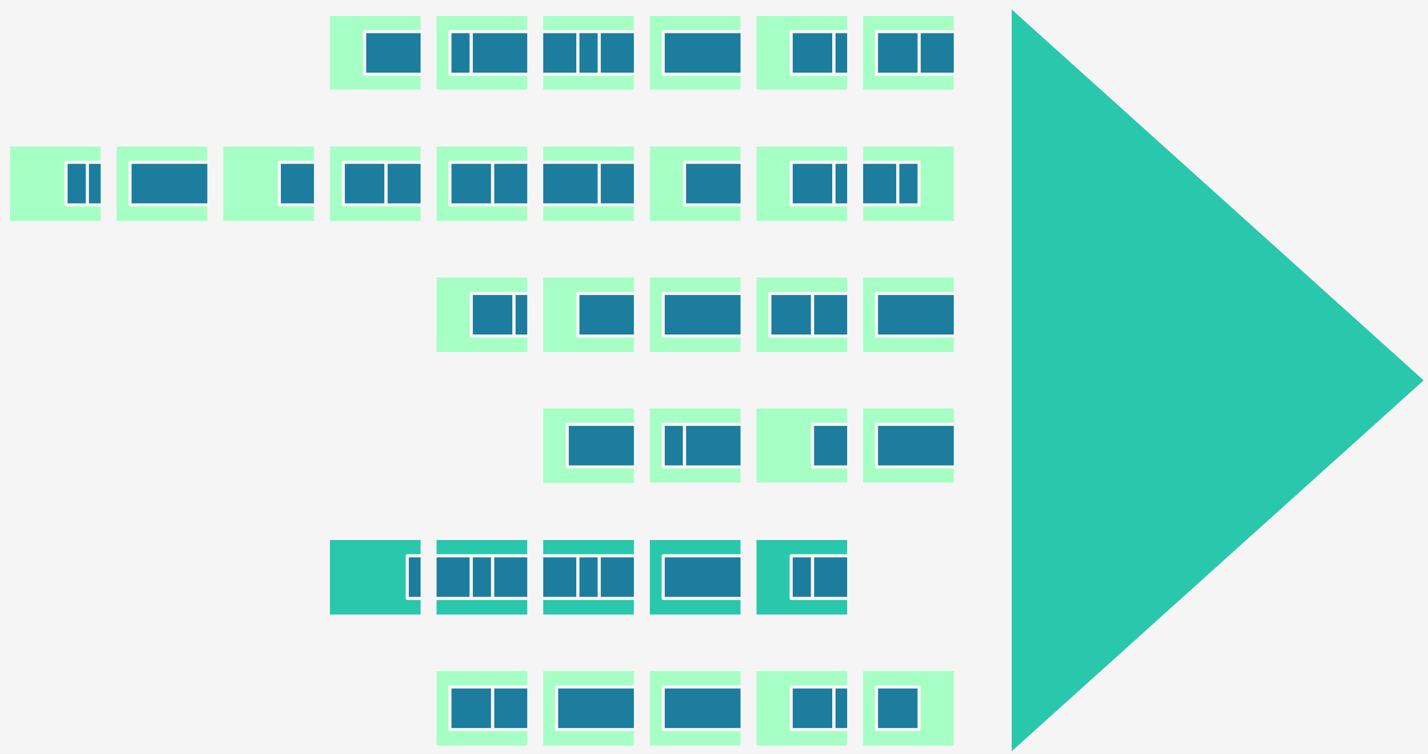
CAMP-MALLOC



CAMP-MALLOC

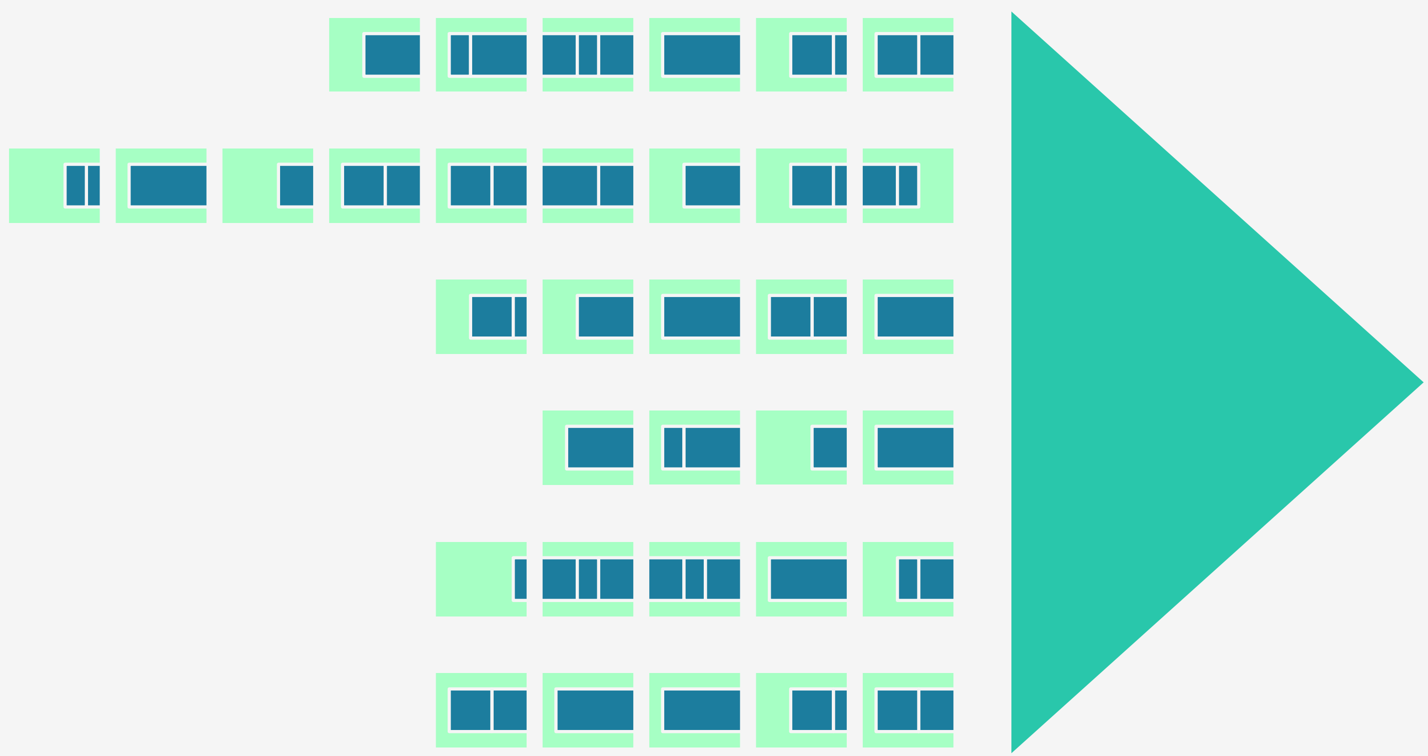


CAMP-MALLOC

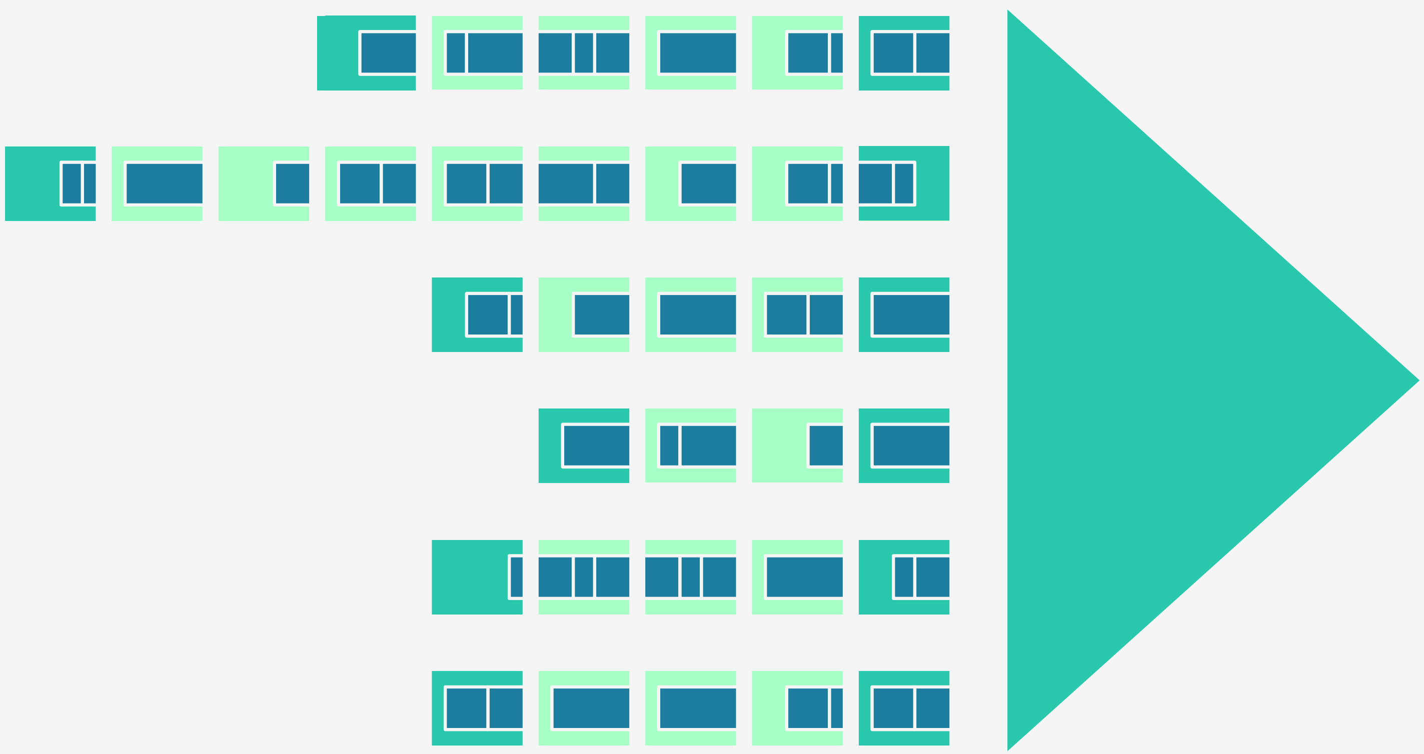


CAMP-MALLOC

CAMP-MALLOC



CAMP-MALLOC



fragmentation $\leq$ 2 (num queues) (block size) + (num blocks) (max item size)

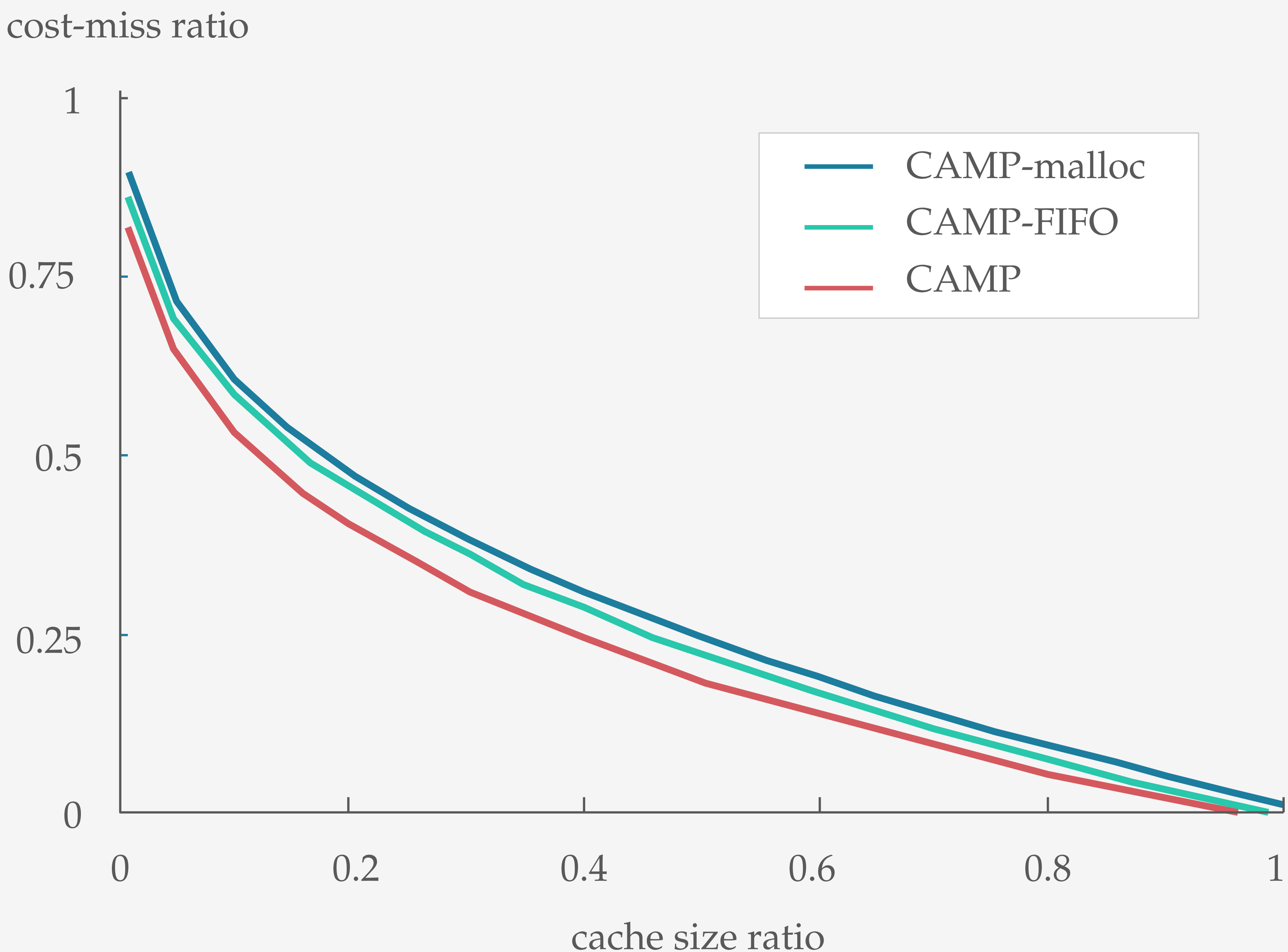
CAMP-MALLOC

is competitive if memory augmented

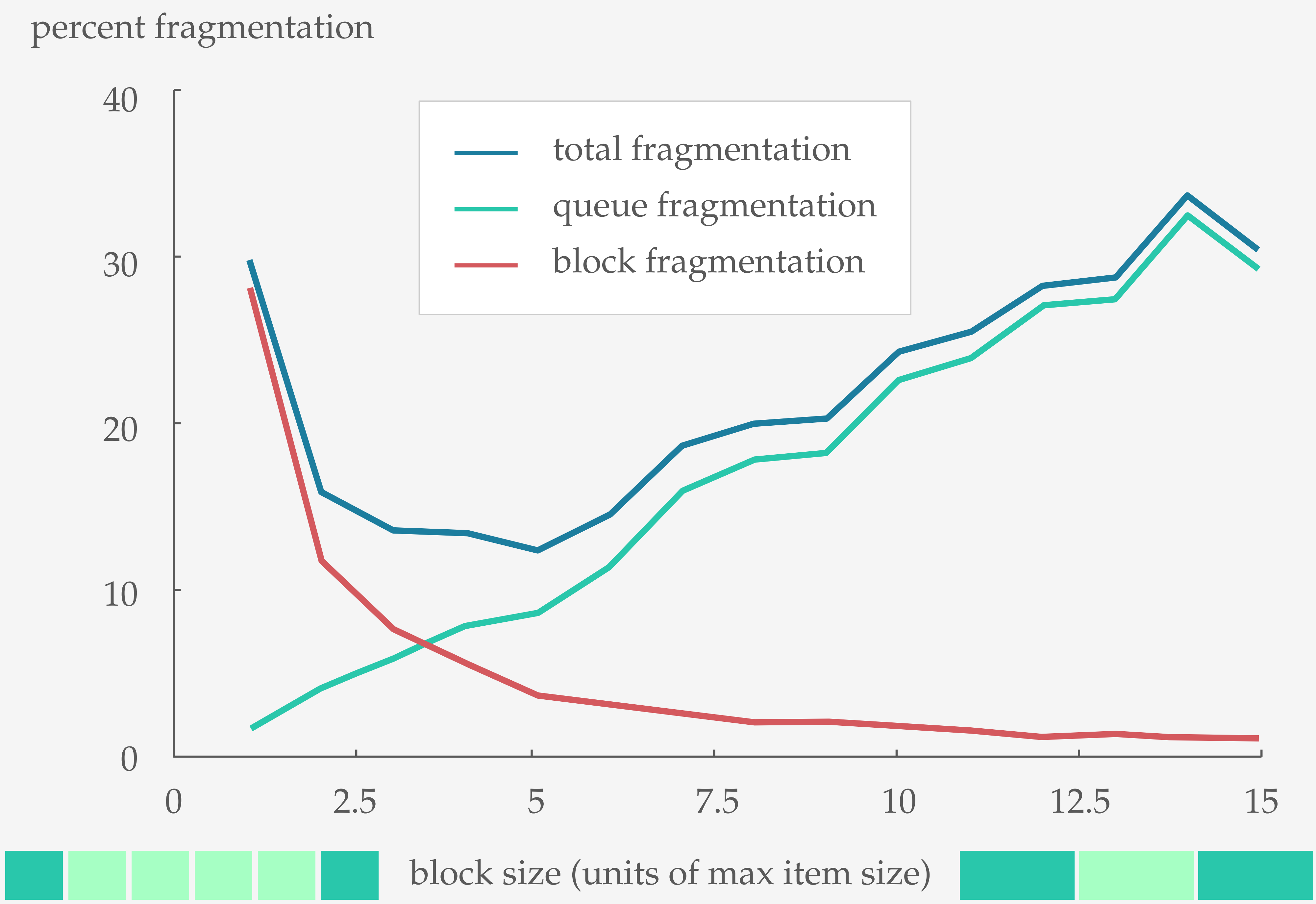
$$\text{if} \quad \begin{array}{c}\text{OPT's}\\\text{cache size}\end{array} \leq \begin{array}{c}\text{C-M's}\\\text{cache size}\end{array} - \begin{array}{c}\text{fragmentation}\\\text{bound}\end{array}$$

$$\text{then} \quad \text{cost(C-M)} \leq \frac{\begin{array}{c}\text{C-M's}\\\text{cache size}\end{array}}{\text{min item size}} \text{cost(OPT)}$$

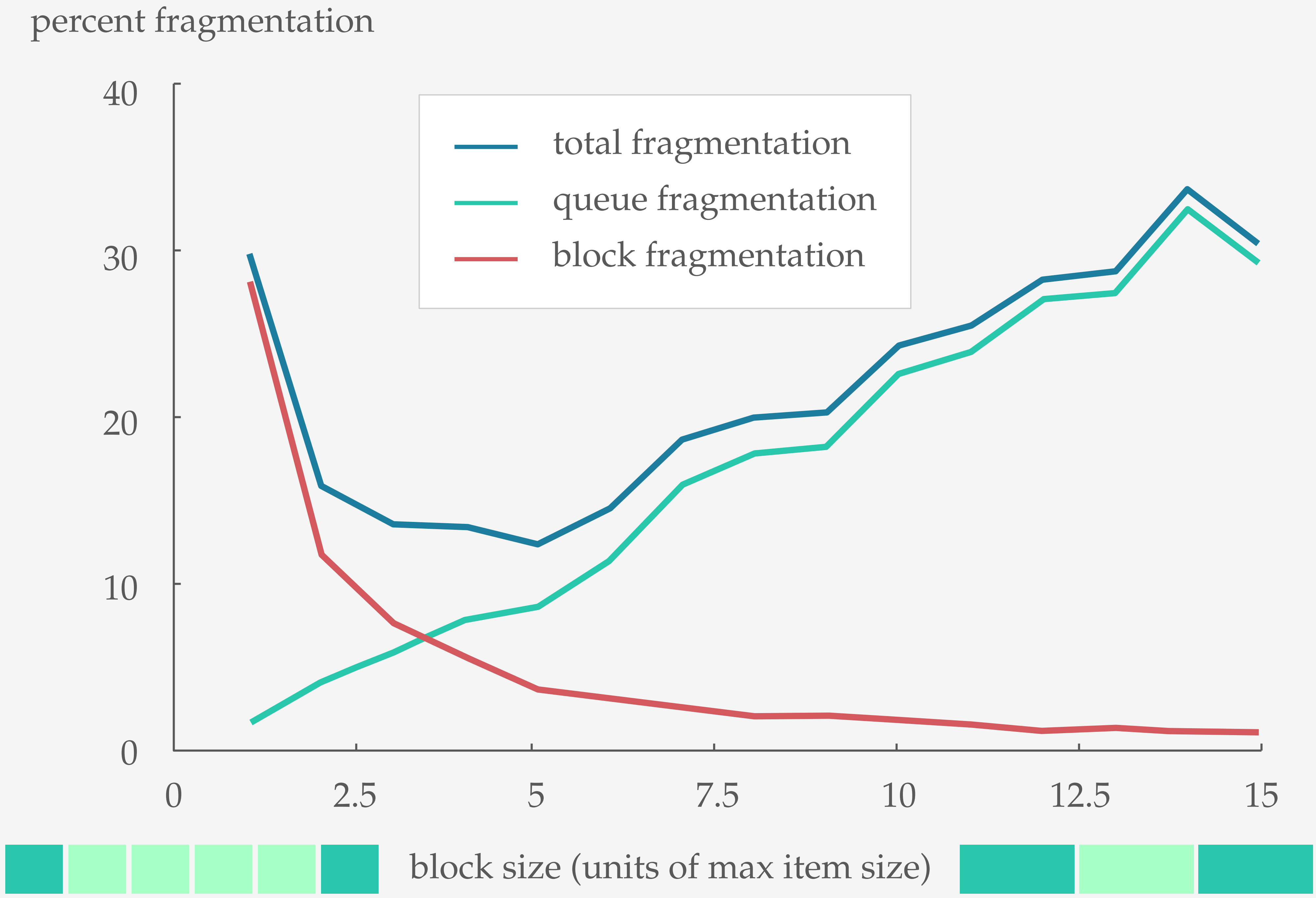
fragmentation $\leq$ 2 (num queues) (block size) + (num blocks) (max item size)

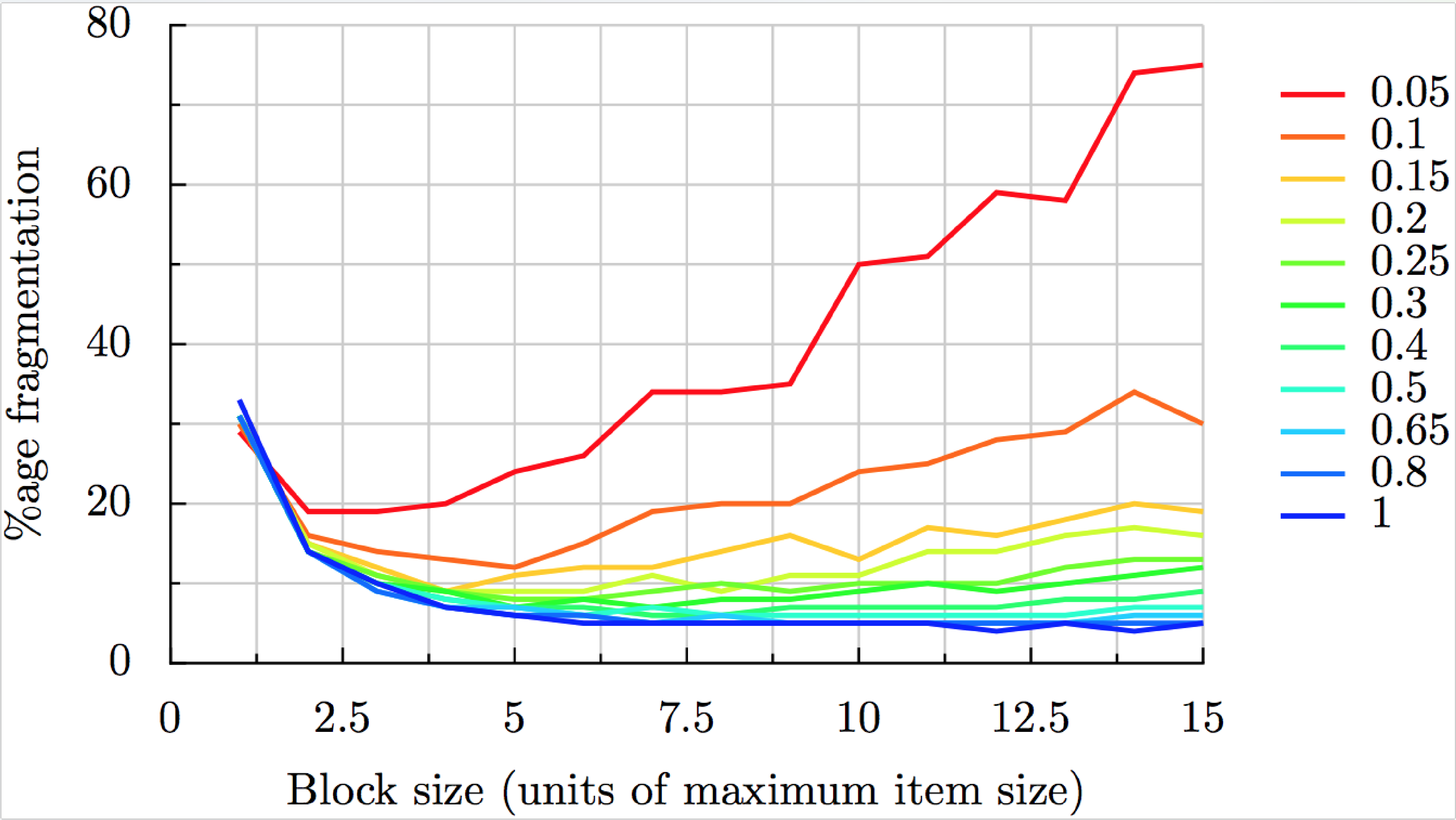


cost-miss ratio
1
0.75
0.5
0.25
0
0
0.2
0.4
0.6
0.8
1
cache size ratio
CAMP-malloc
CAMP-FIFO
CAMP

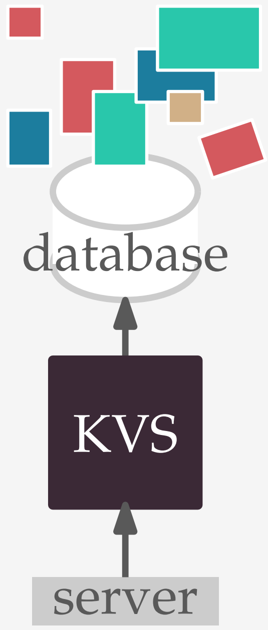


percent fragmentation
40
30
20
10
0
total fragmentation
queue fragmentation
block fragmentation
0
2.5
5
7.5
10
12.5
15
block size (units of max item size)


percent fragmentation
40
30
20
10
0
total fragmentation
queue fragmentation
block fragmentation
0
2.5
5
7.5
10
12.5
15
block size (units of max item size)



80
60
40
20
0
0
2.5
5
7.5
10
12.5
15
%age fragmentation
Block size (units of maximum item size)
0.05
0.1
0.15
0.2
0.25
0.3
0.4
0.5
0.65
0.8
1



EVICTION POLICY

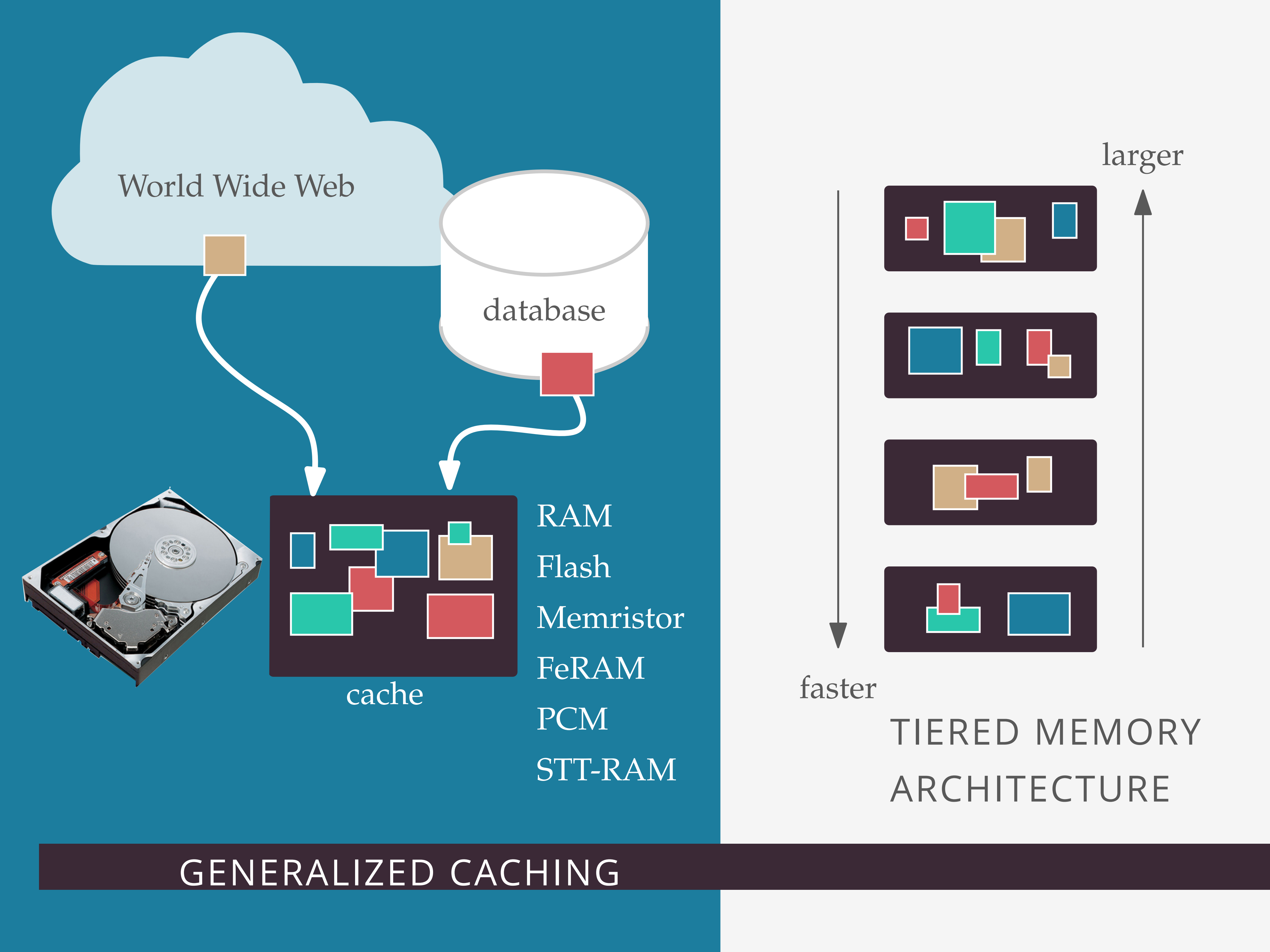
GDS → CAMP

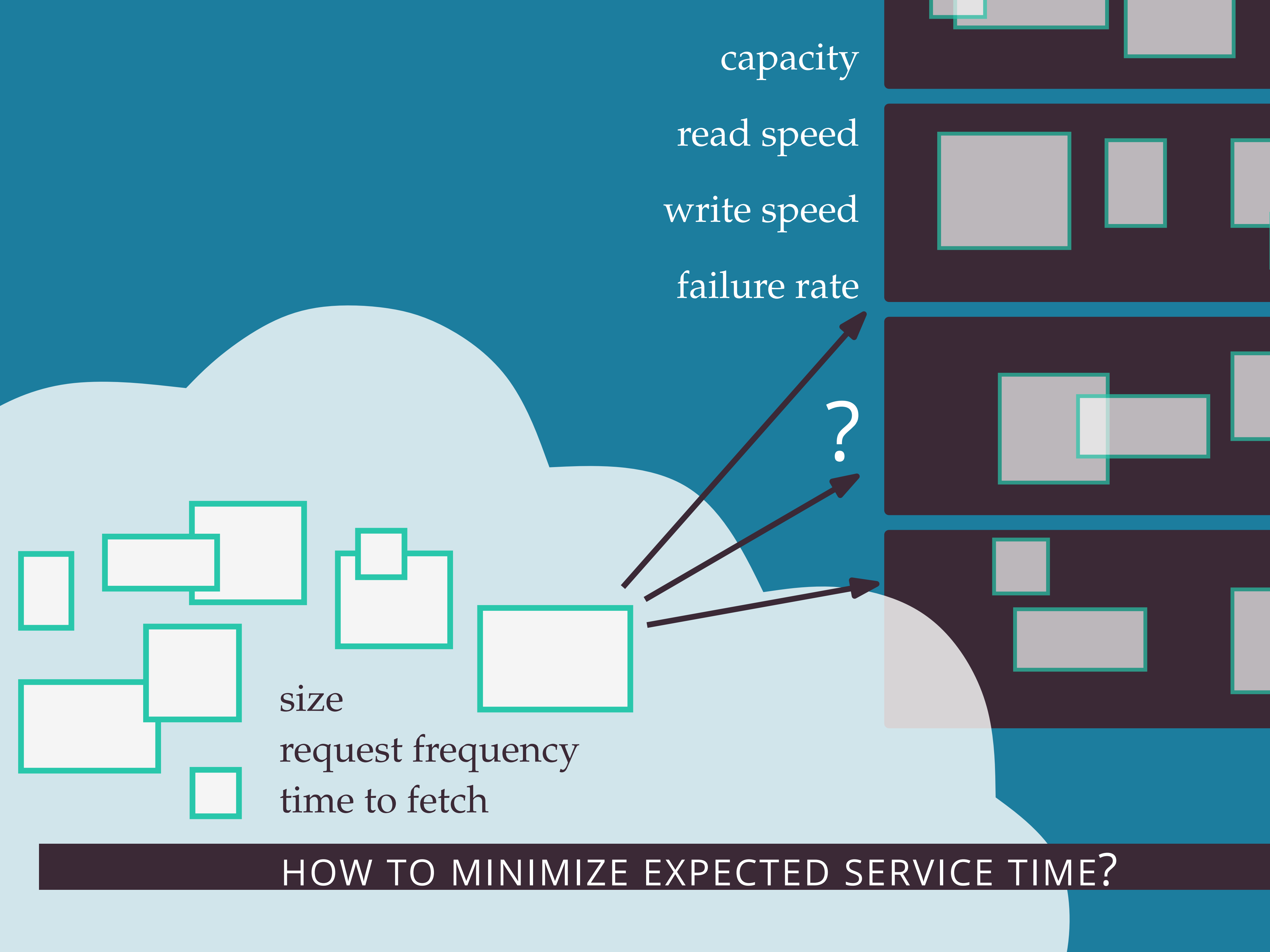
PLACEMENT POLICY

generalized caching → managed memory caching

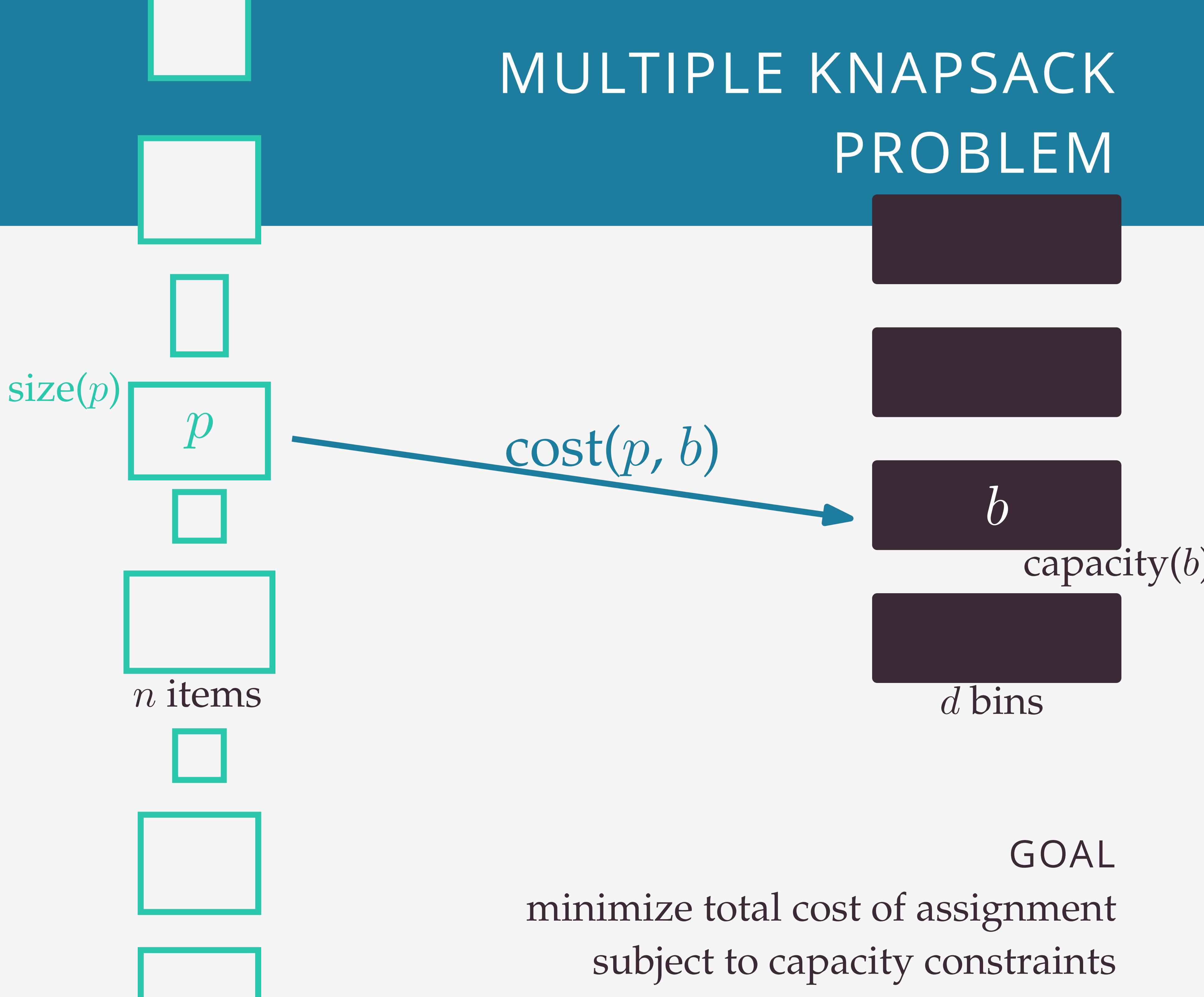
MEMORY HIERARCHY

2-level cache → multi-level cache


World Wide Web
database
RAM
Flash
Memristor
FeRAM
PCM
STT-RAM
cache
larger
faster
TIERED MEMORY
ARCHITECTURE
GENERALIZED CACHING


capacity
read speed
write speed
failure rate
?
size
request frequency
time to fetch
HOW TO MINIMIZE EXPECTED SERVICE TIME?

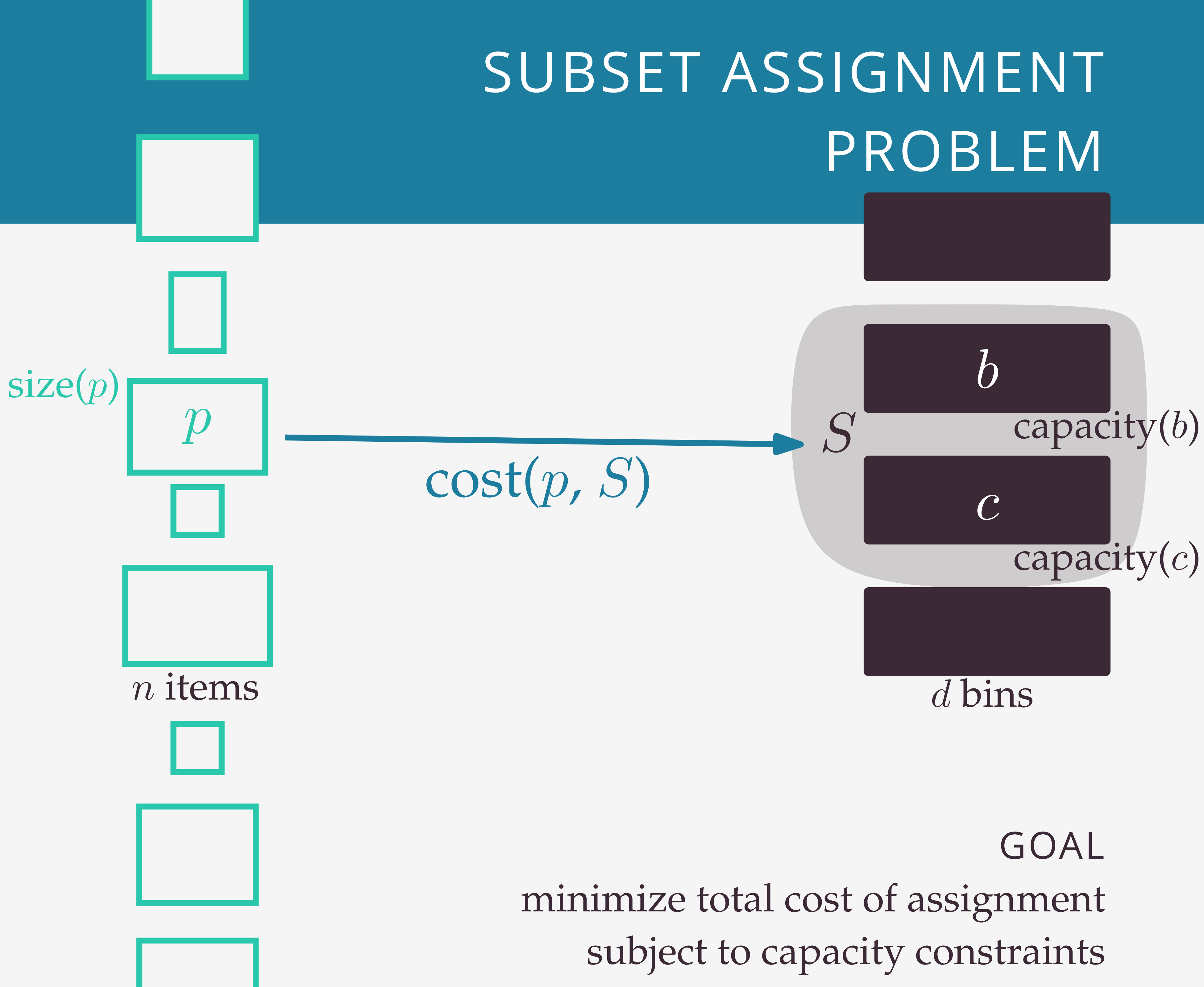
MULTIPLE KNAPSACK PROBLEM



GOAL

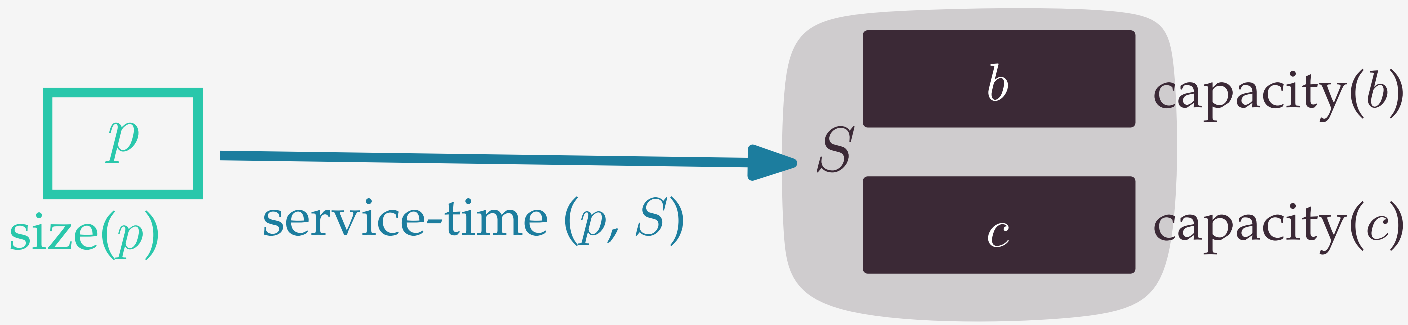
minimize total cost of assignment
subject to capacity constraints

SUBSET ASSIGNMENT PROBLEM

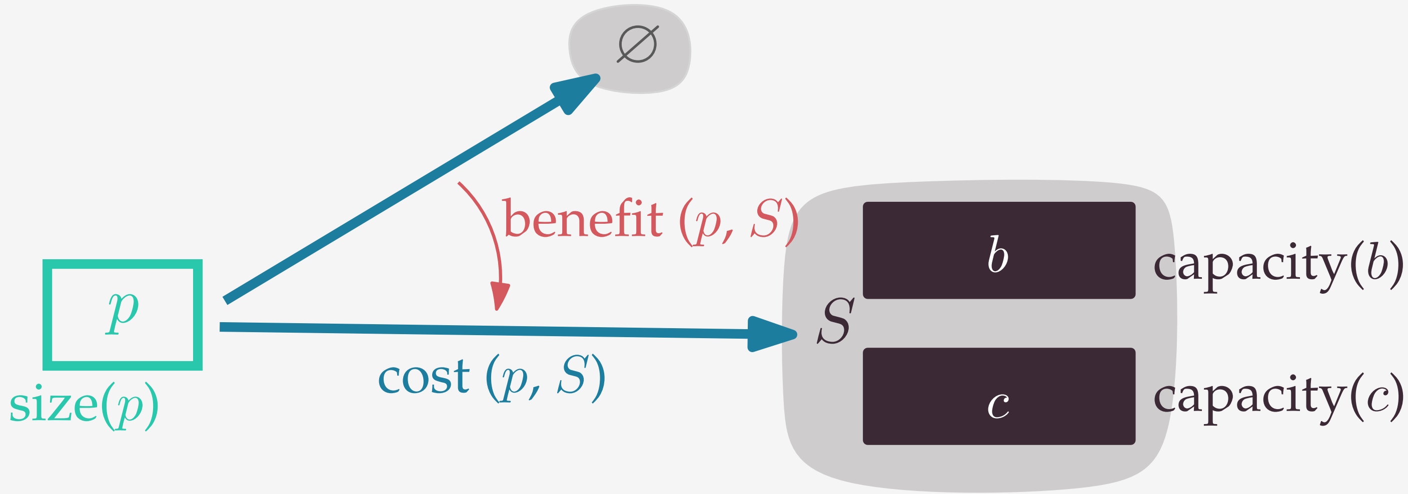


GOAL

minimize total cost of assignment
subject to capacity constraints



$$
\begin{aligned}
\text{service-time}(p,S) \;=\;& \text{read-frequency}(p)\;\;\text{read-time}(p,S)\\
+\;& \text{write-frequency}(p)\;\;\text{write-time}(p,S)\\
+\;& \sum_{F\subseteq S} \text{fail-freq}(F)\;\Big(\text{read-time}(p, S\setminus F)\\
&\qquad + \text{write-time}(p, S\cap F)\Big)
\end{aligned}
$$



cache configuration

$$\text{maximize} \sum_{p,S} \text{benefit}(p,S)\, x(p,S)$$

$$\sum_{S} x(p,S) = 1$$

$$\sum_{p,S} \text{price}(p,S) x(p,S) \leq \text{budget}$$

$$x = 0, 1$$

subset assignment

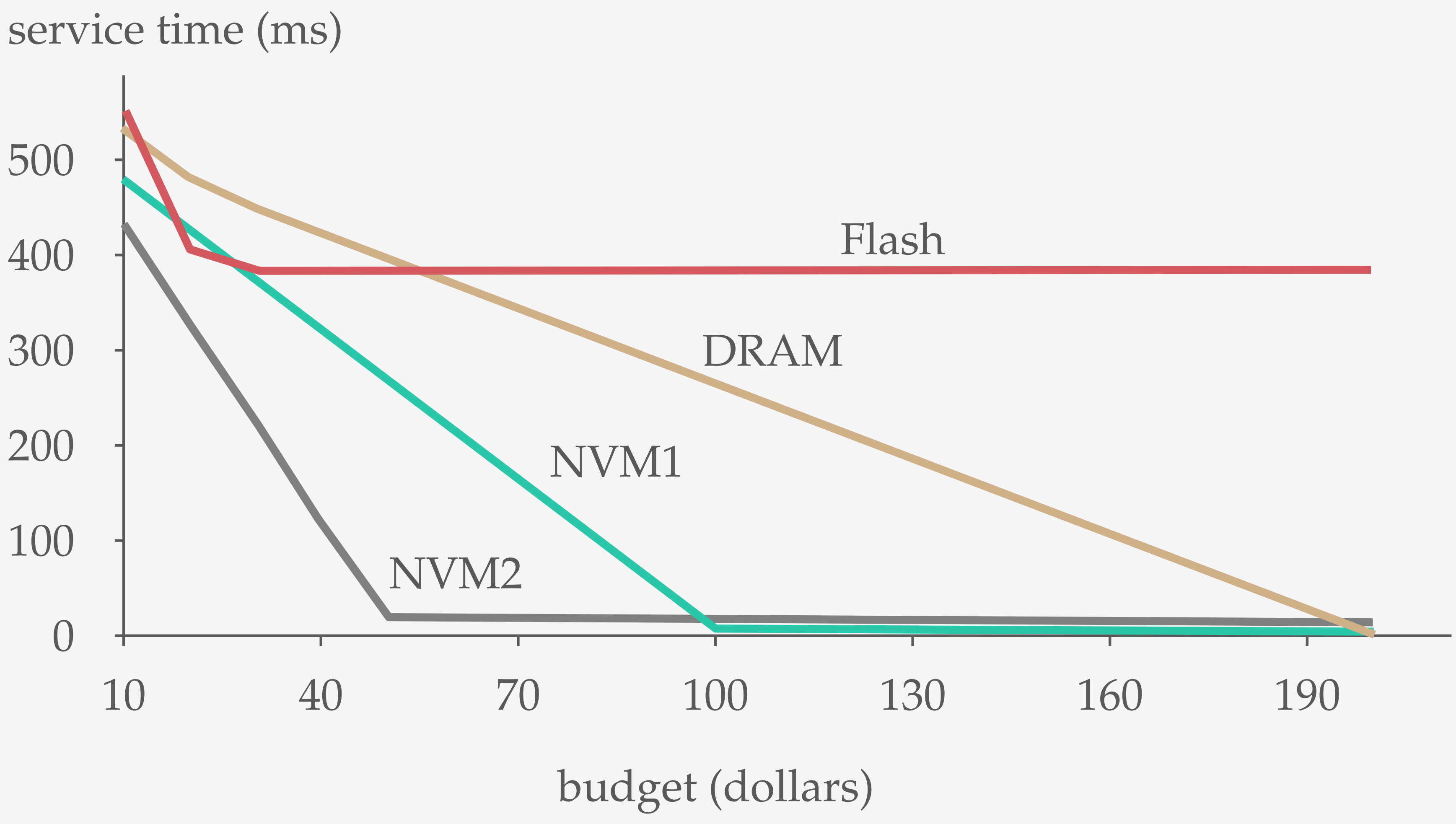
$$\text{minimize} \sum_{p,S} \text{cost}(p,S)\, x(p,S)$$

$$\sum_{S} x(p,S) = \text{size}(p)$$

$$\sum_{p,S \ni b} x(p,S) \leq \text{capacity}(b)$$

$$x(p,S) = 0, \text{size}(p)$$

CACHE CONFIGURATION



SUBSET ASSIGNMENT

HAVE $d \ll n$

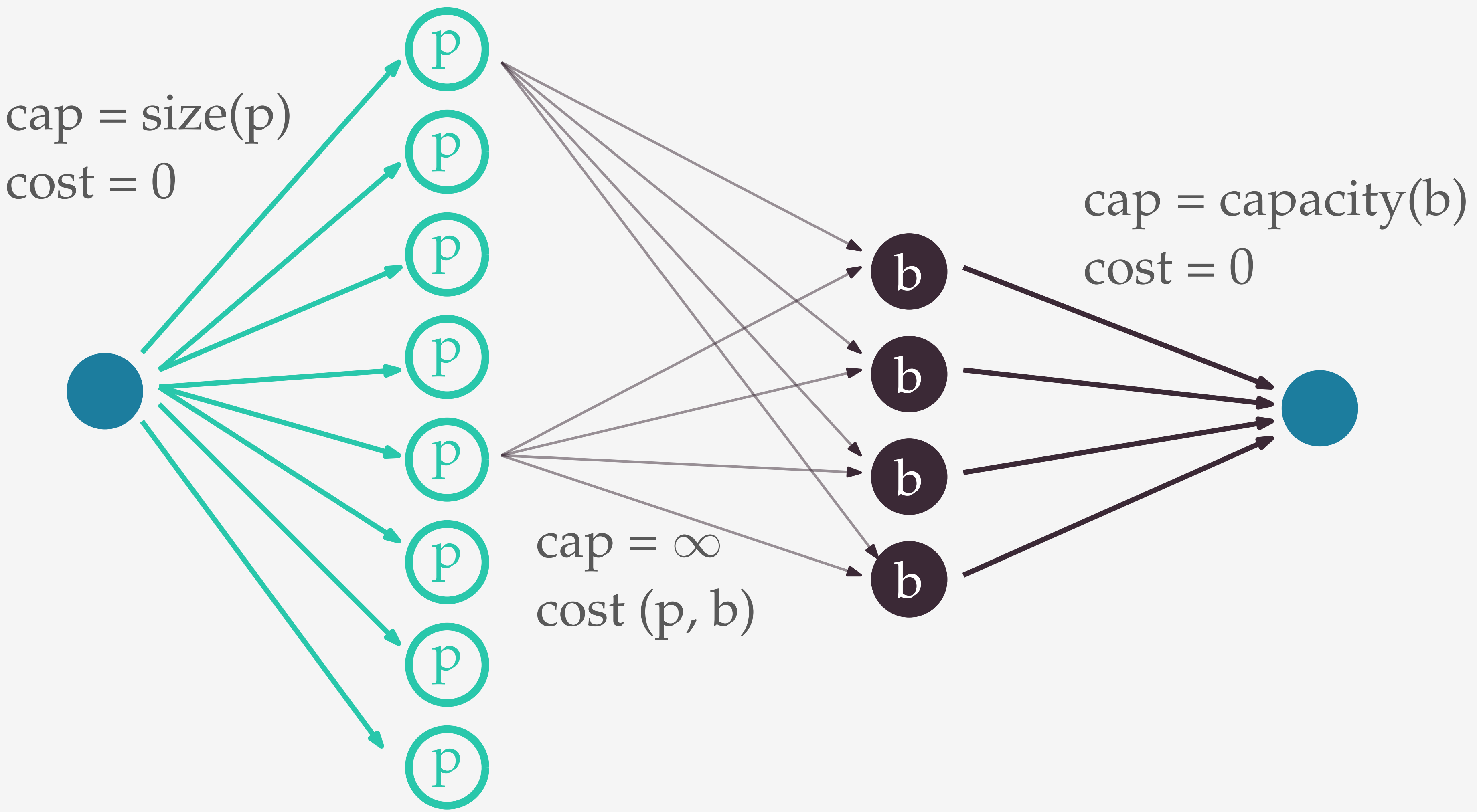
sol to LP relaxation has few fractional assignments

GOAL solve LP relaxation in f(d) poly(n)

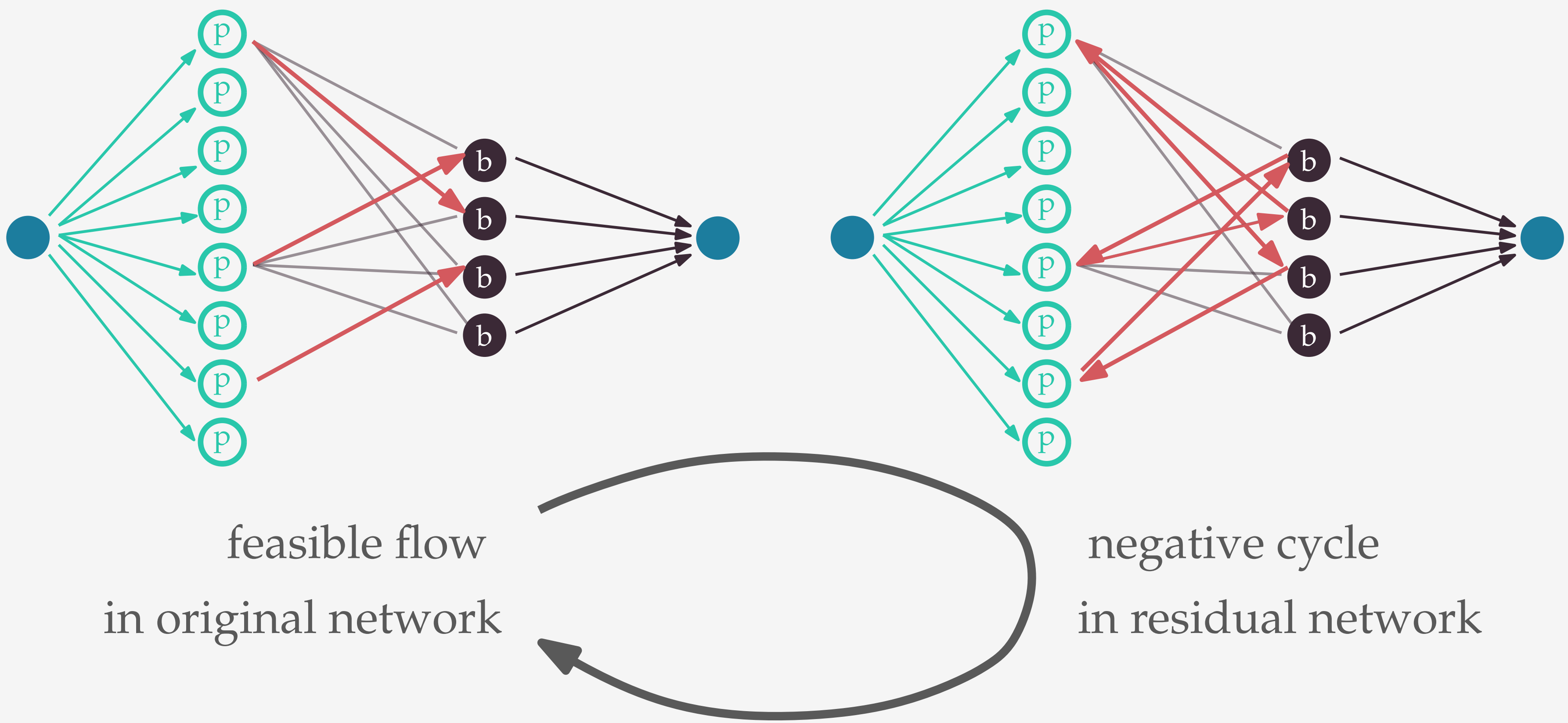
1. cycle canceling algorithm

2. simplex algorithm

MIN COST FLOW



1. cycle canceling algorithm



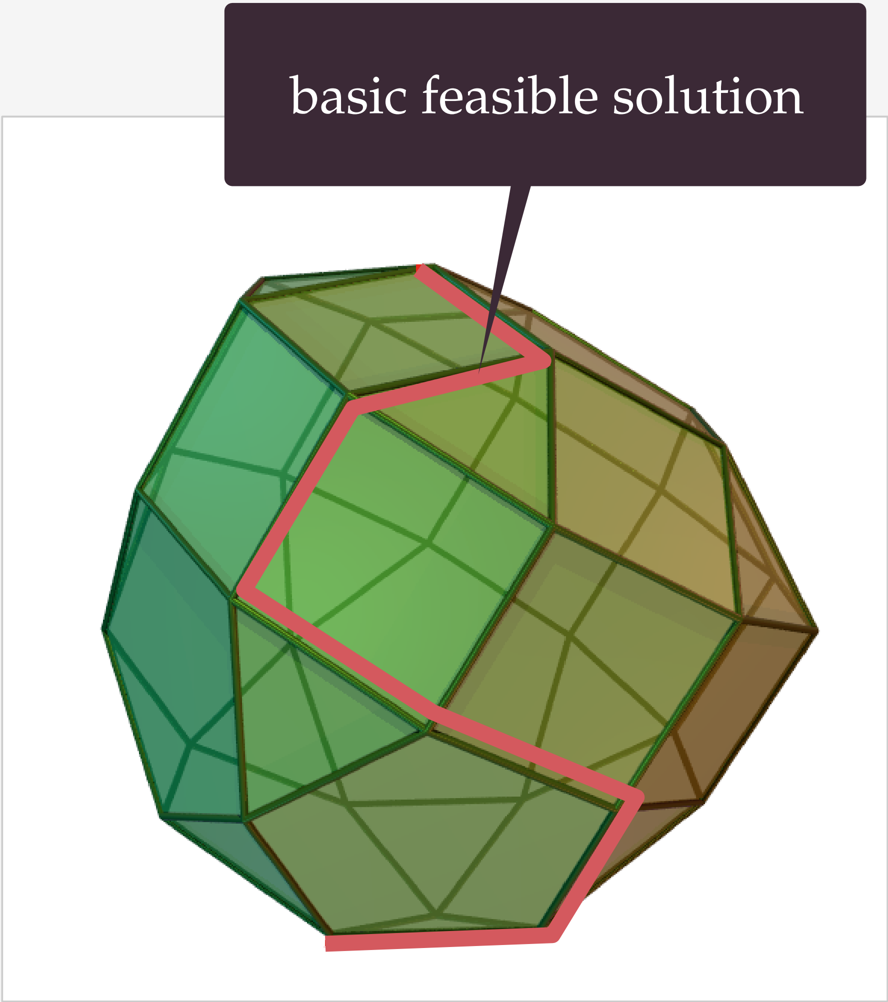
“cycle” in subset assignment problem

augmentation $S_i \xrightarrow{p_i} T_i$

such that $$\sum_i \alpha_i \overrightarrow{S_i T_i} = \vec{0}$$

cost difference (negative) $$\sum_i \alpha_i \ (\text{cost}(p_i, T_i) - \text{cost}(p_i, S_i))$$

2. simplex algorithm



BASIC FEASIBLE ASSIGNMENT

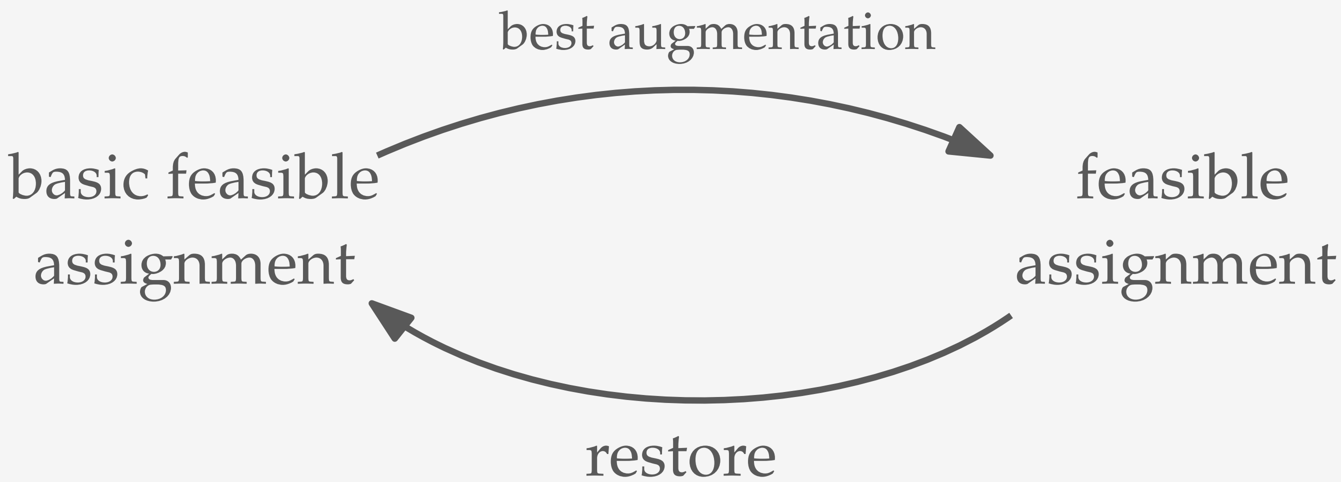
< 2d fractional assignments

bound granularity of vars

$$x(p, S) = \frac{k}{\ell}$$

$< d^{d/2}$

ALGORITHM


best augmentation
basic feasible assignment
feasible assignment
restore

ALGORITHM

preprocessing $\sum_i \alpha_i \overrightarrow{S_i T_i} = \vec{0}$

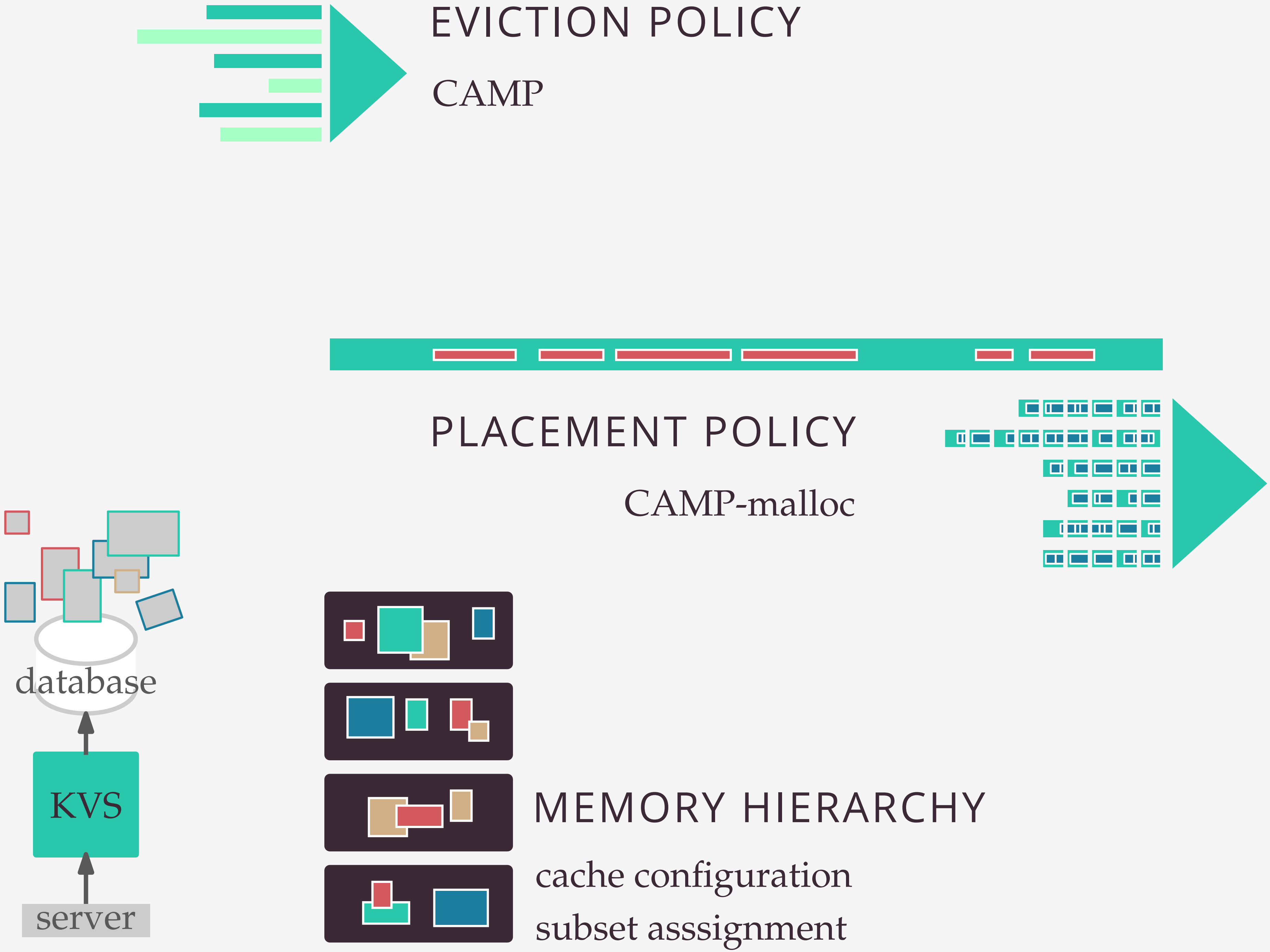
best augmentation

basic feasible assignment $\longrightarrow$ feasible assignment

feasible assignment $\longrightarrow$ basic feasible assignment

restore

time $O(\exp(d(d+1)\text{poly}(d)\;\; n\log(n)\log(nC)\log(S))$


EVICTION POLICY
CAMP
PLACEMENT POLICY
CAMP-malloc
database
KVS
server
MEMORY HIERARCHY
cache configuration
subset assignment

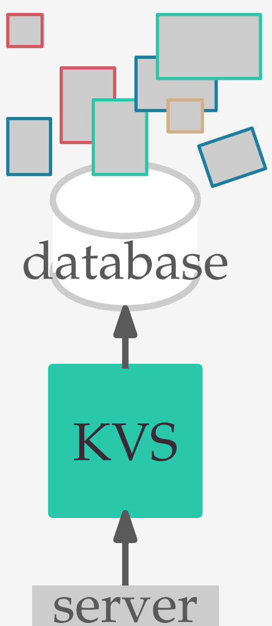
EVICTION POLICY

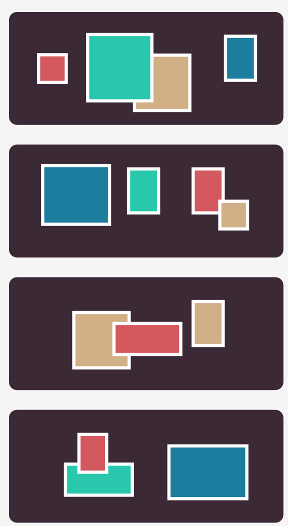
CAMP

PLACEMENT POLICY

CAMP-malloc

LRU-FIFO hybrid?

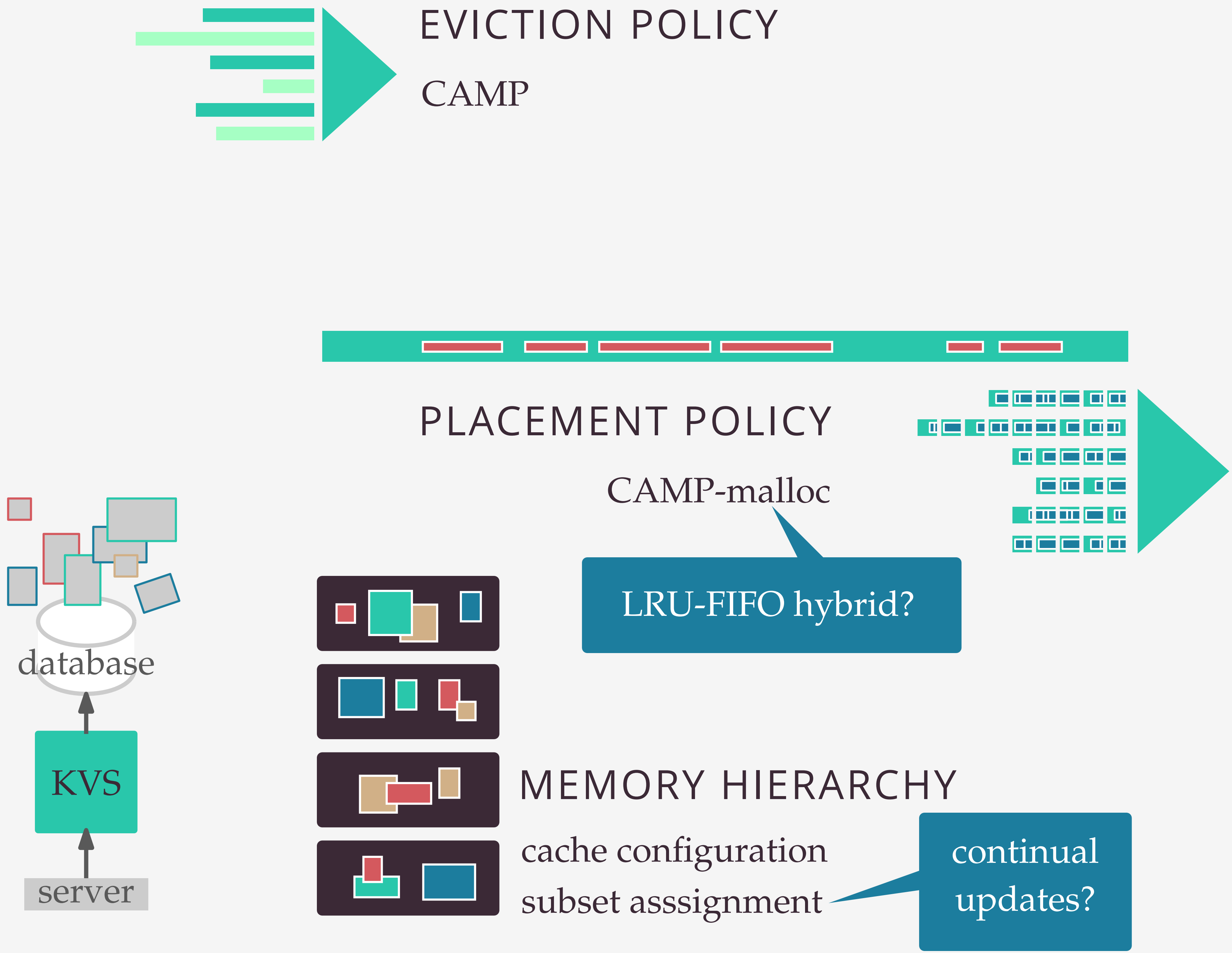


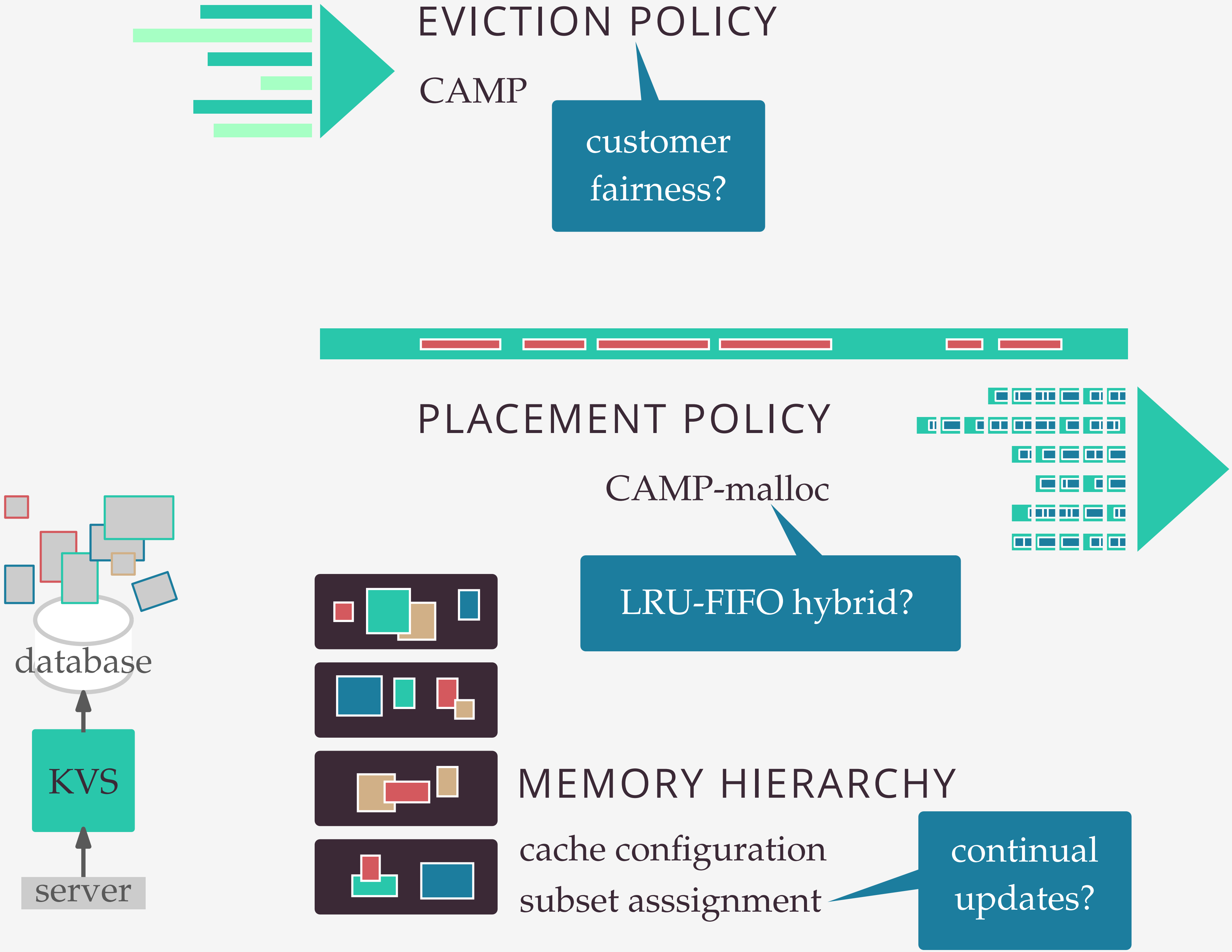


MEMORY HIERARCHY

cache configuration

subset assignment

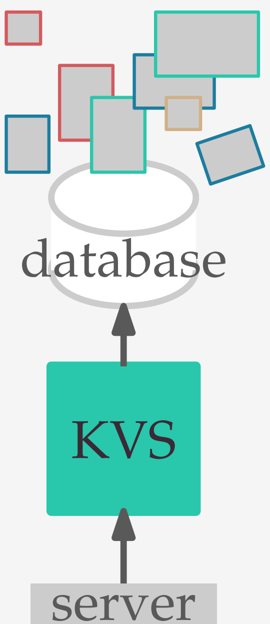

EVICTION POLICY
CAMP
PLACEMENT POLICY
CAMP-malloc
LRU-FIFO hybrid?
database
KVS
server
MEMORY HIERARCHY
cache configuration
subset assignment
continual updates?

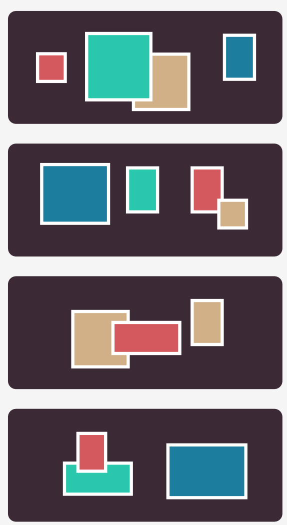

EVICTION POLICY
CAMP
customer fairness?
PLACEMENT POLICY
CAMP-malloc
LRU-FIFO hybrid?
database
KVS
server
MEMORY HIERARCHY
cache configuration
subset assignment
continual updates?

CAMP: a cost adaptive multi-queue eviction policy for key-value stores. Shahram Ghandeharizadeh, Sandy Irani, Jenny Lam, and Jason Yap. In Proceedings of the 15th International Middleware Conference, 2014.

A demonstration of KOSAR: an elastic, scalable, highly available SQL middleware. Shahram Ghandeharizadeh, Connor Gorman, Sandy Irani, Shiva Jahangiri, Jenny Lam, Hieu Nguyen, Ryan Tani and Jason Yap, Middleware 2014.

Cache replacement with memory allocation. Shahram Ghandeharizadeh, Sandy Irani, and Jenny Lam. In Proceedings of the 17th Workshop on Algorithm Engineering and Experiments (ALENEX), 2015.





Memory hierarchy design for caching middleware in the age of NVM. Shahram Ghandeharizadeh, Sandy Irani, and Jenny Lam. (in submission)

The subset assignment problem for data placement in caches. Shahram Ghandeharizadeh, Sandy Irani, and Jenny Lam. (in submission)