Middleware 2014 — December 12, 2014

CAMP

a Cost Adaptive Multiqueue eviction Policy for key-value stores

University of California, Irvine
University of Southern California

Jenny Lam
Shahram Ghandeharizadeh
Sandy Irani
Jason Yap

the problem

amazon

facebook

twitter GitHub

LinkedIn blizzard
interactive

digg

Dynamo

amazon

facebook

twitter GitHub

LinkedIn blizzard interactive

digg

Dynamo

Memcached

amazon

facebook

twitter GitHub

LinkedIn

blizzard interactive

digg

Dynamo

Memcached

amazon

facebook

twitter

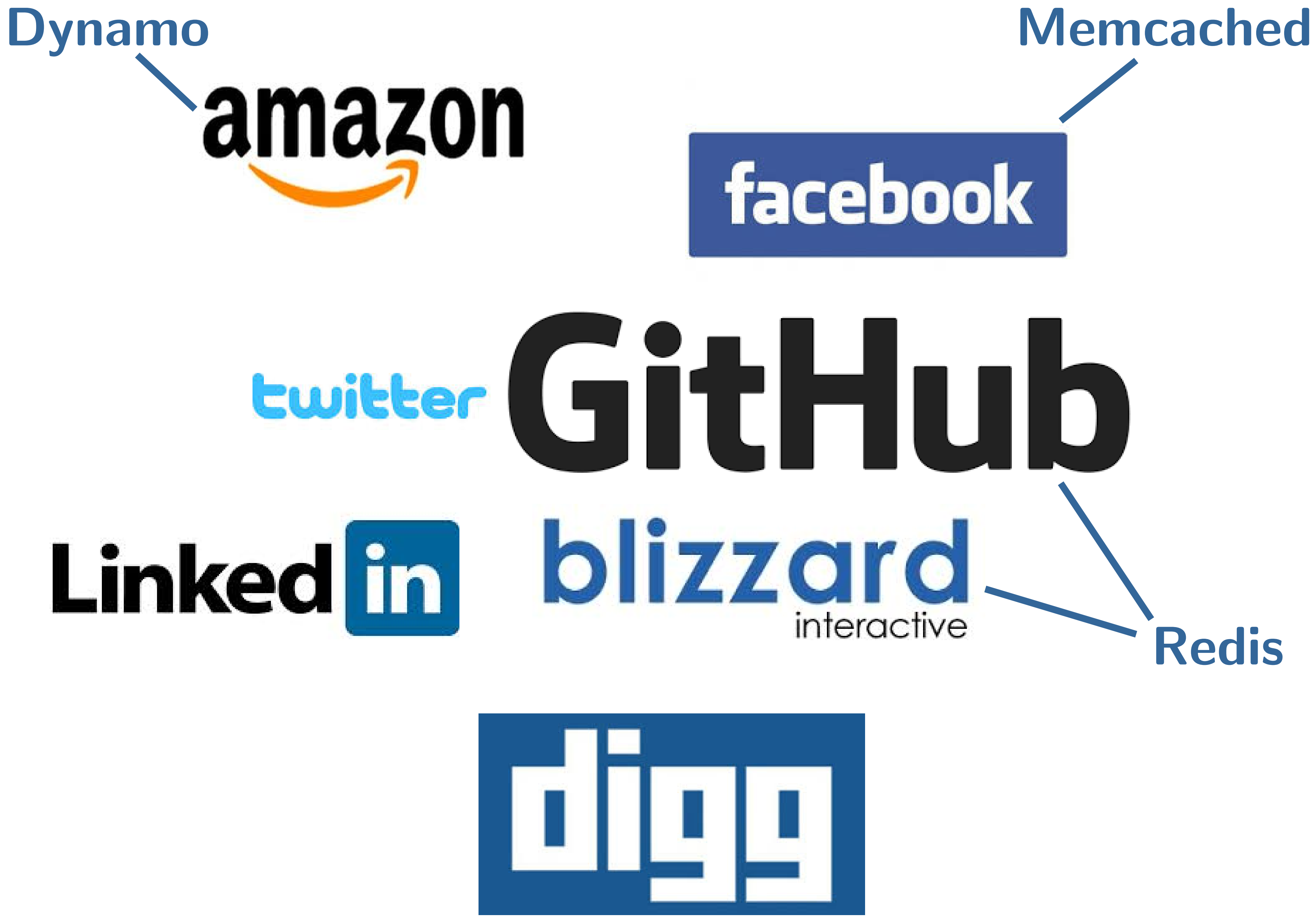
GitHub

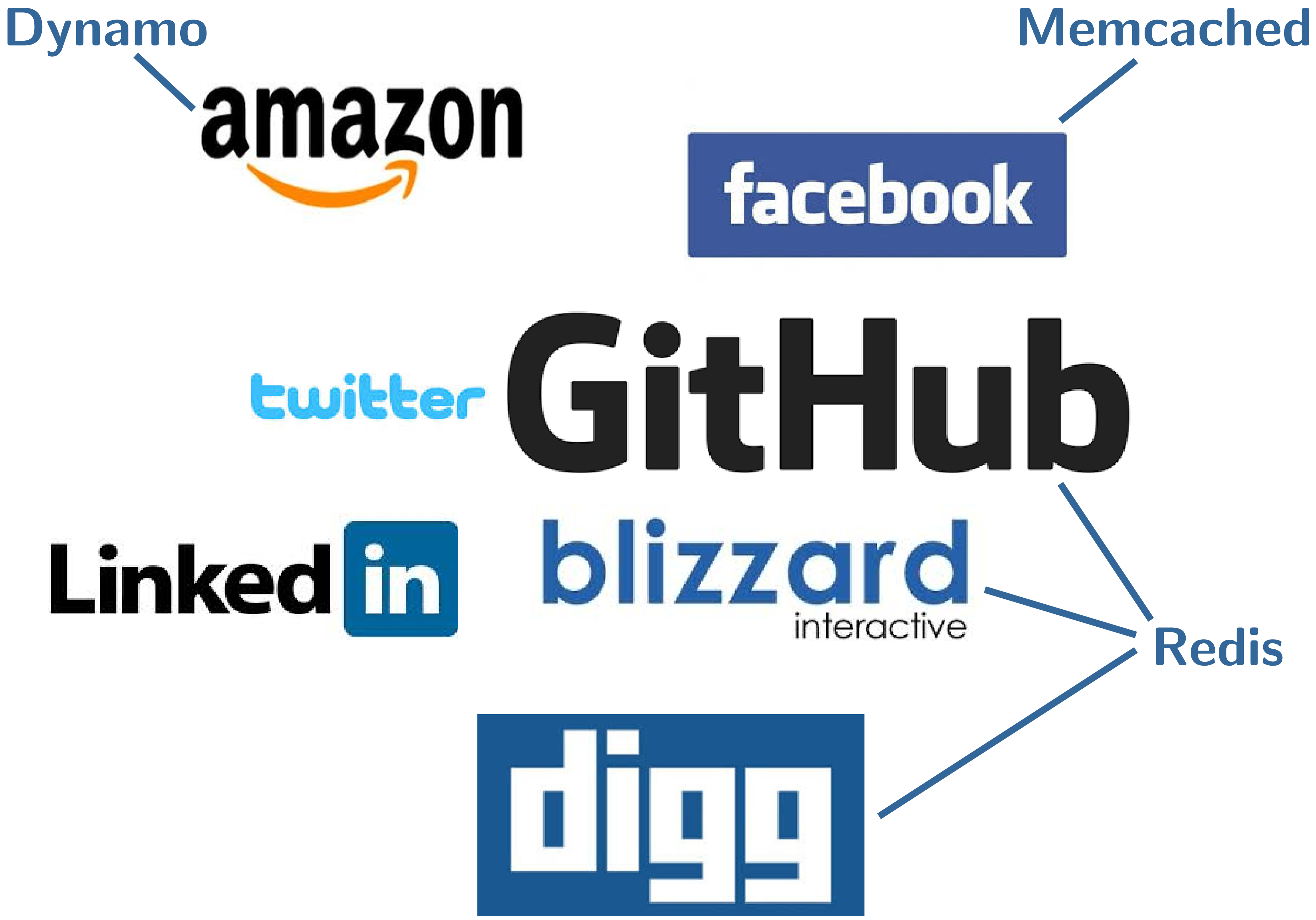
Linked in

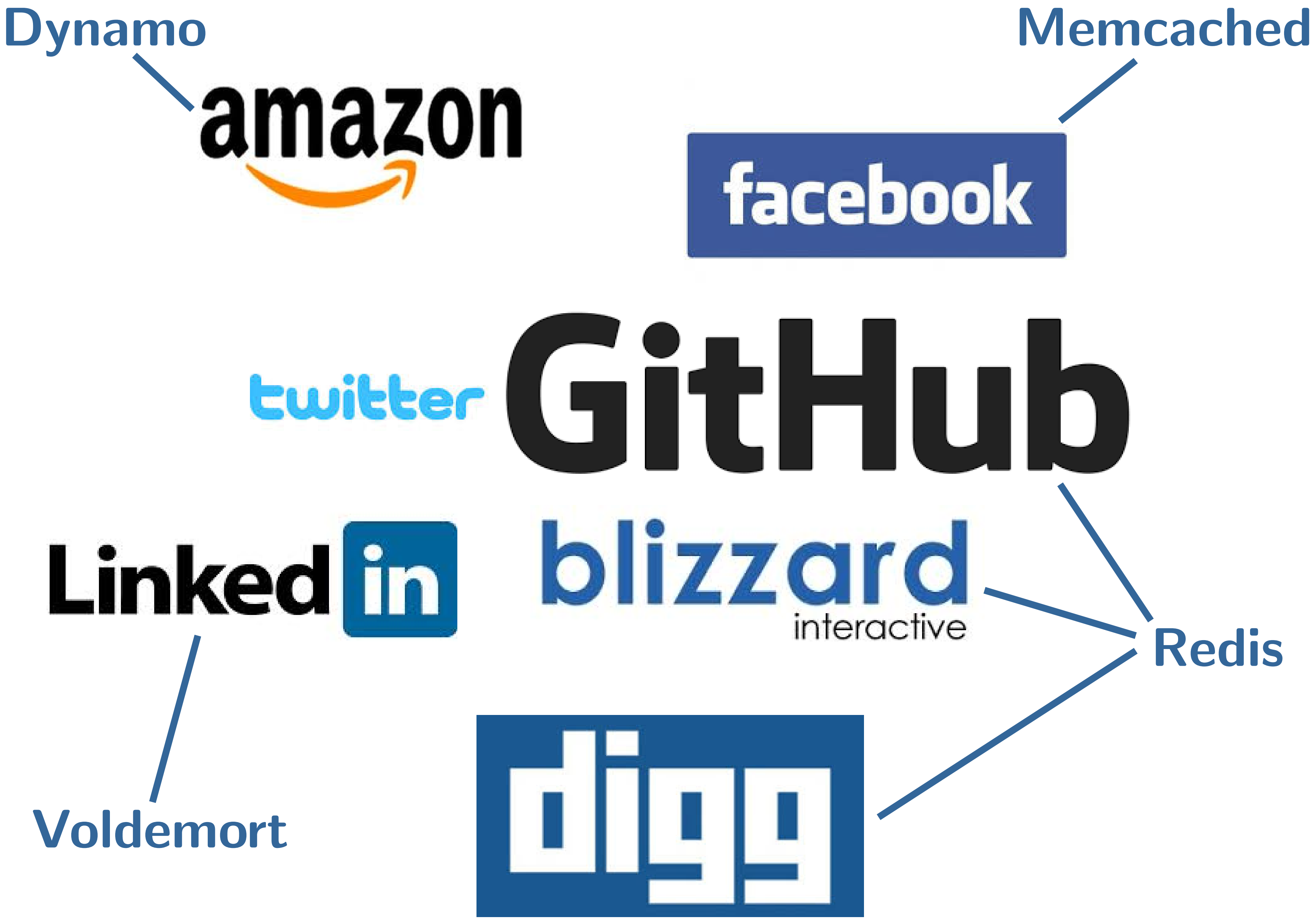
blizzard interactive

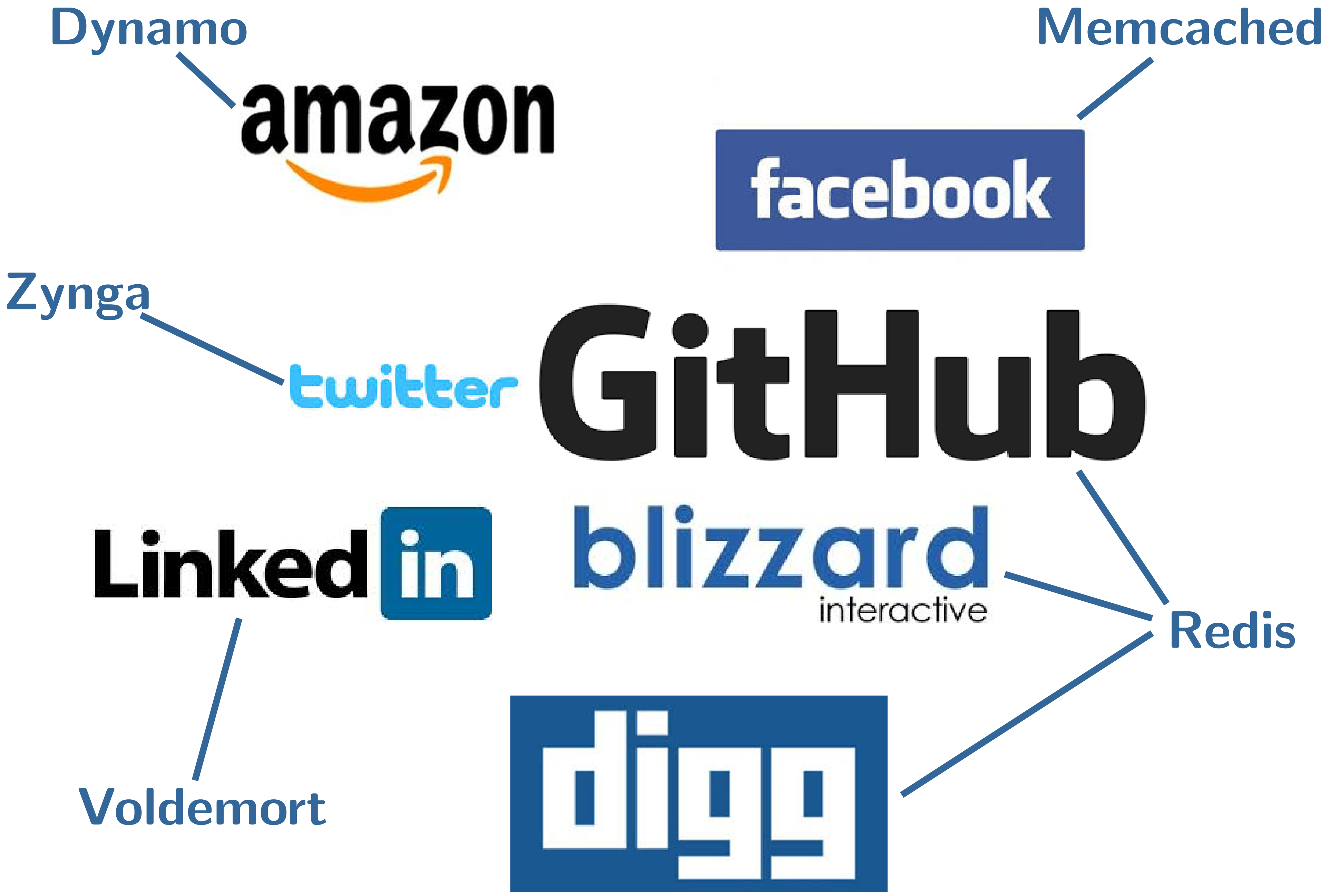
Redis

digg


Dynamo
Memcached
amazon
facebook
twitter
GitHub
LinkedIn
blizzard interactive
Redis
digg


Dynamo
amazon
Memcached
facebook
twitter
GitHub
LinkedIn
blizzard
interactive
Redis
digg


Dynamo
amazon
Memcached
facebook
twitter
GitHub
Linked in
blizzard interactive
Redis
Voldemort
digg


Dynamo
amazon
Memcached
facebook
Zynga
twitter
GitHub
LinkedIn
blizzard interactive
Redis
Voldemort
digg

Dynamo

Memcached

Zynga

?

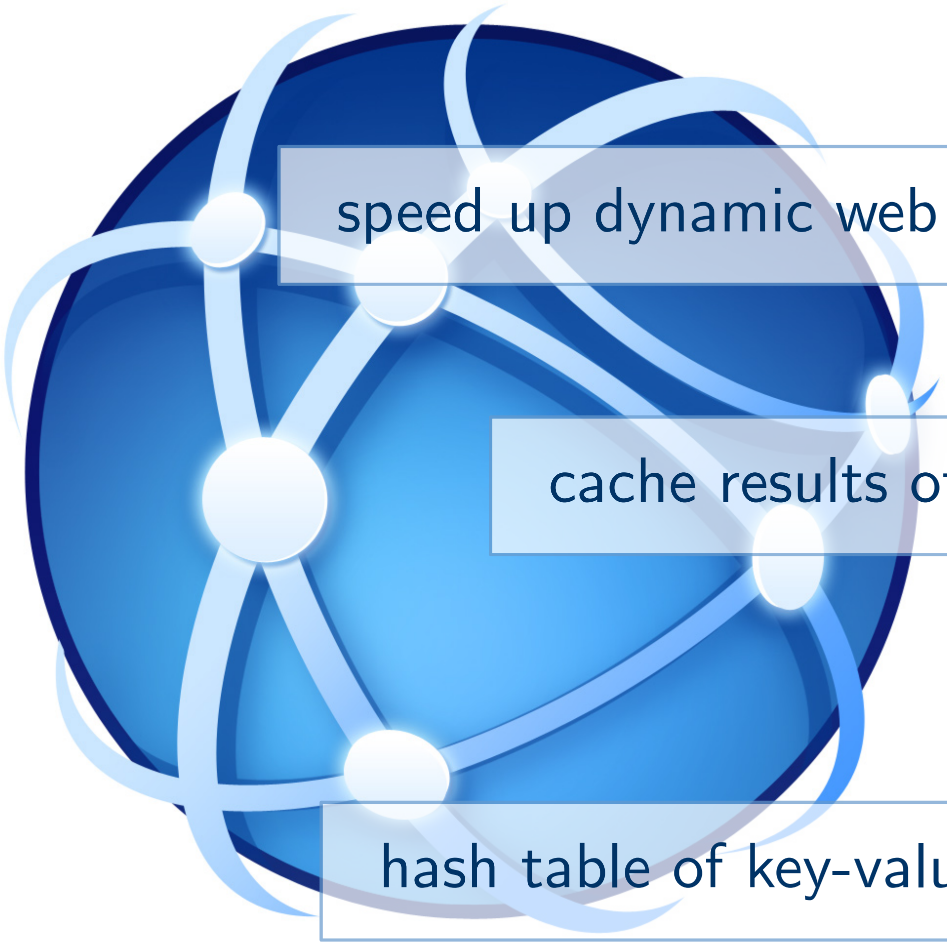
Redis

Voldemort

Dynamo

Memcached

Zynga



speed up dynamic web services

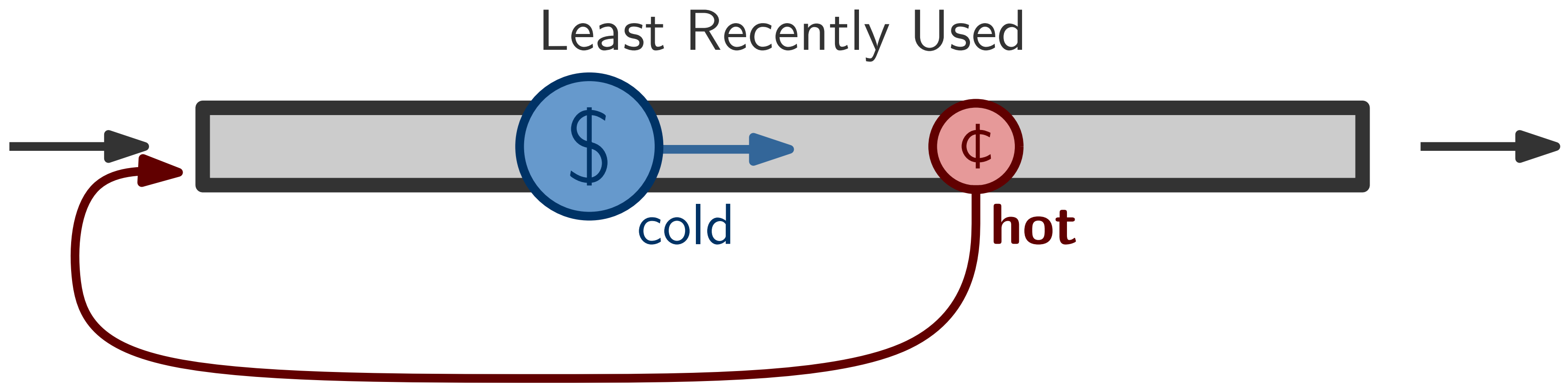
cache results of computation

hash table of key-value pairs

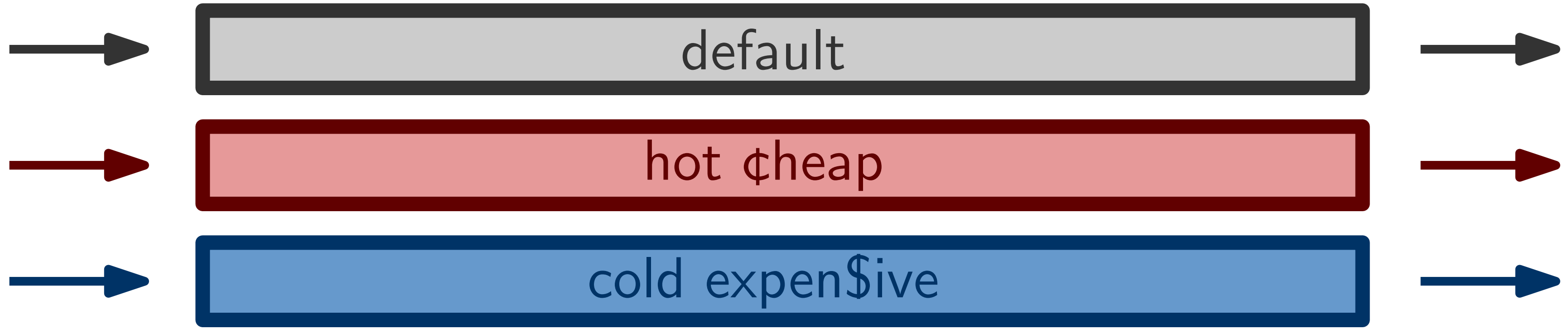
Redis

Voldemort

Least Recently Used

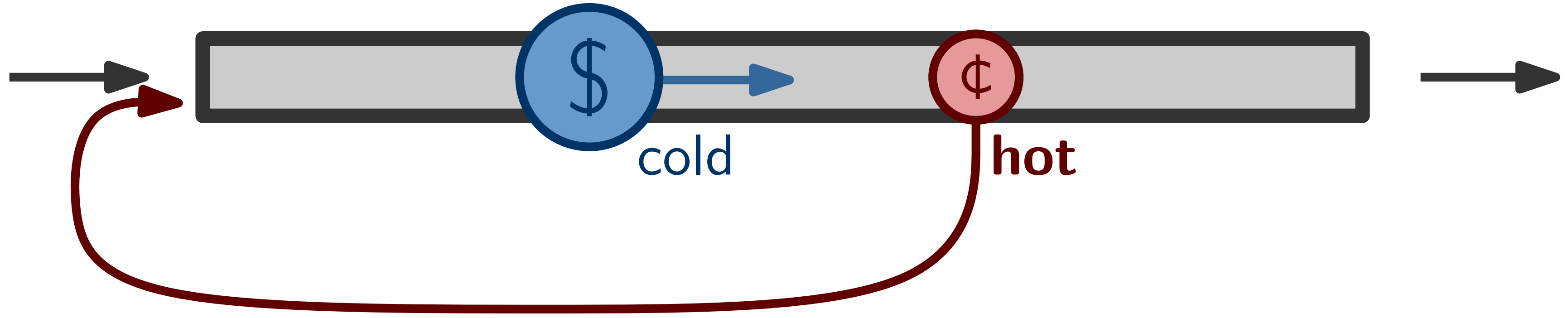

Least Recently Used
$
¢
cold
hot

pooled Least Recently Used

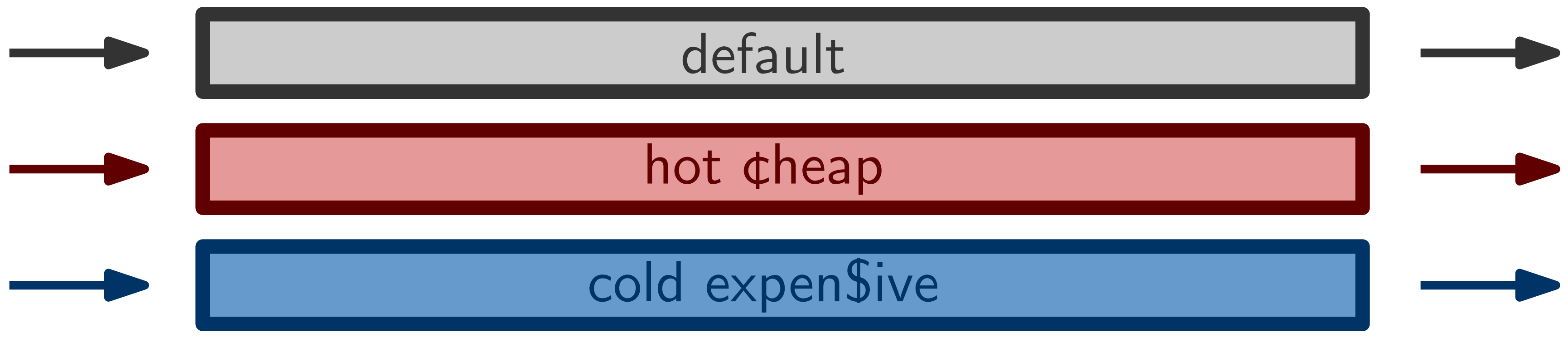


Scaling Memcache at Facebook, Nishtala et al.

Least Recently Used

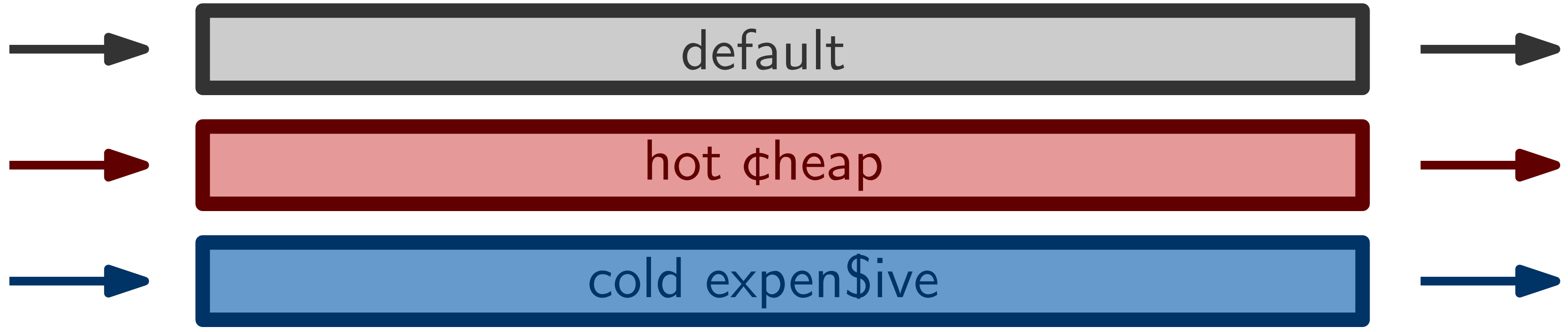


pooled Least Recently Used



Scaling Memcache at Facebook, Nishtala et al.

pooled Least Recently Used



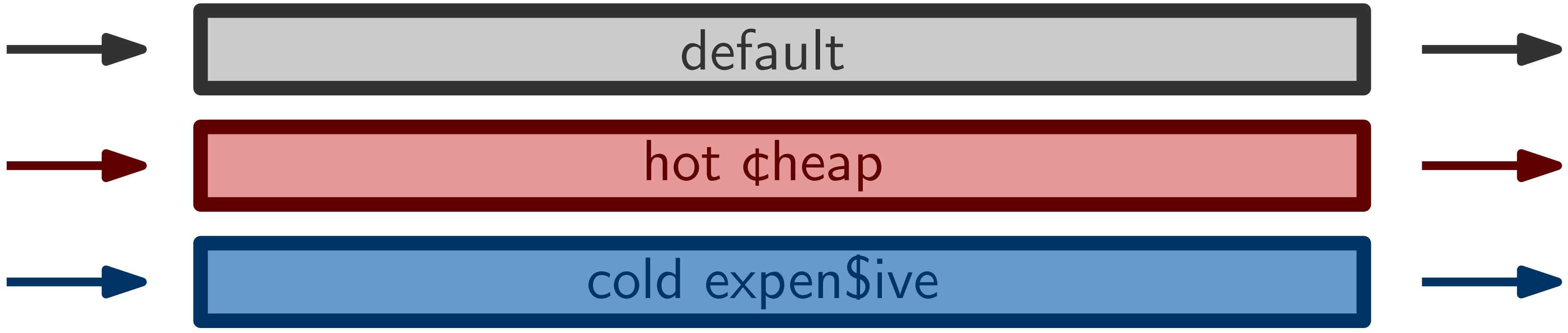
Scaling Memcache at Facebook, Nishtala et al.

assign key-value pairs to pools

assign memory to pools manually and statically

reconfigure when access pattern changes

pooled Least Recently Used



Scaling Memcache at Facebook, Nishtala et al.

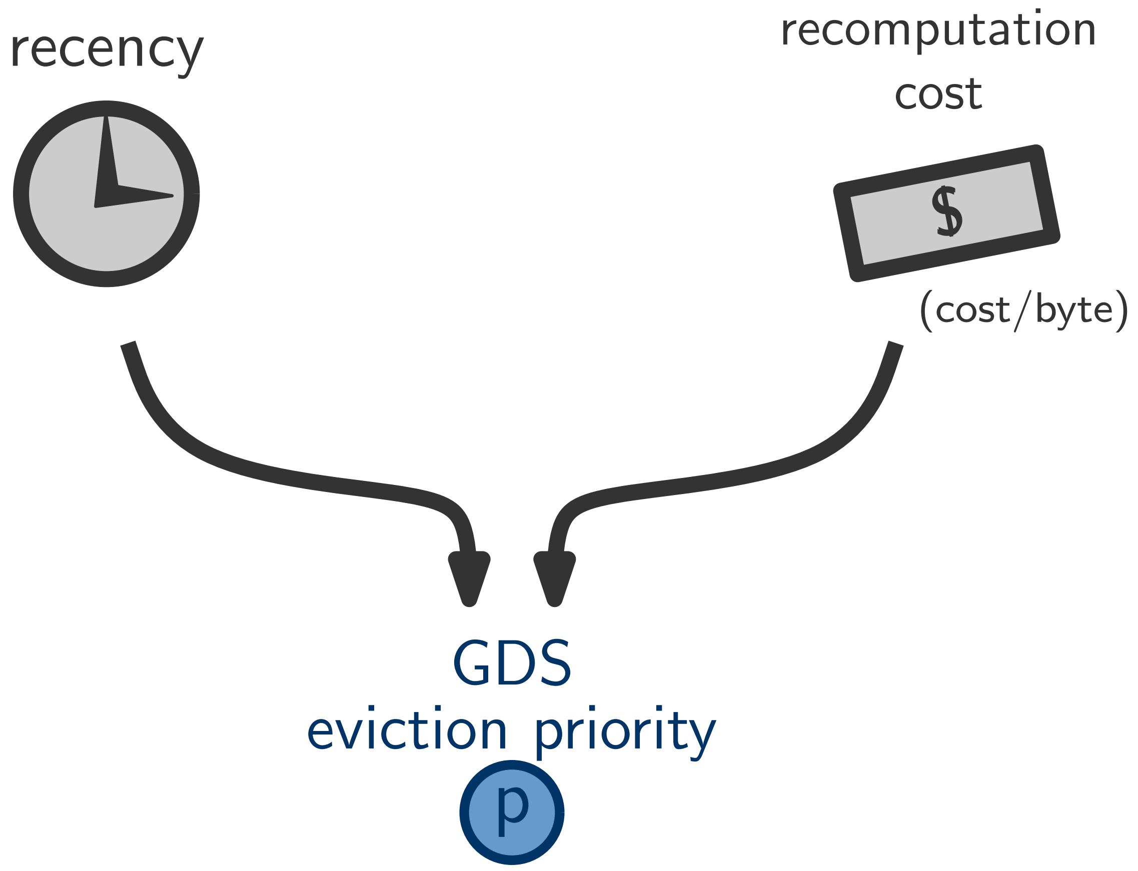
assign key-value pairs to pools

assign memory to pools manually and statically

reconfigure when access pattern changes

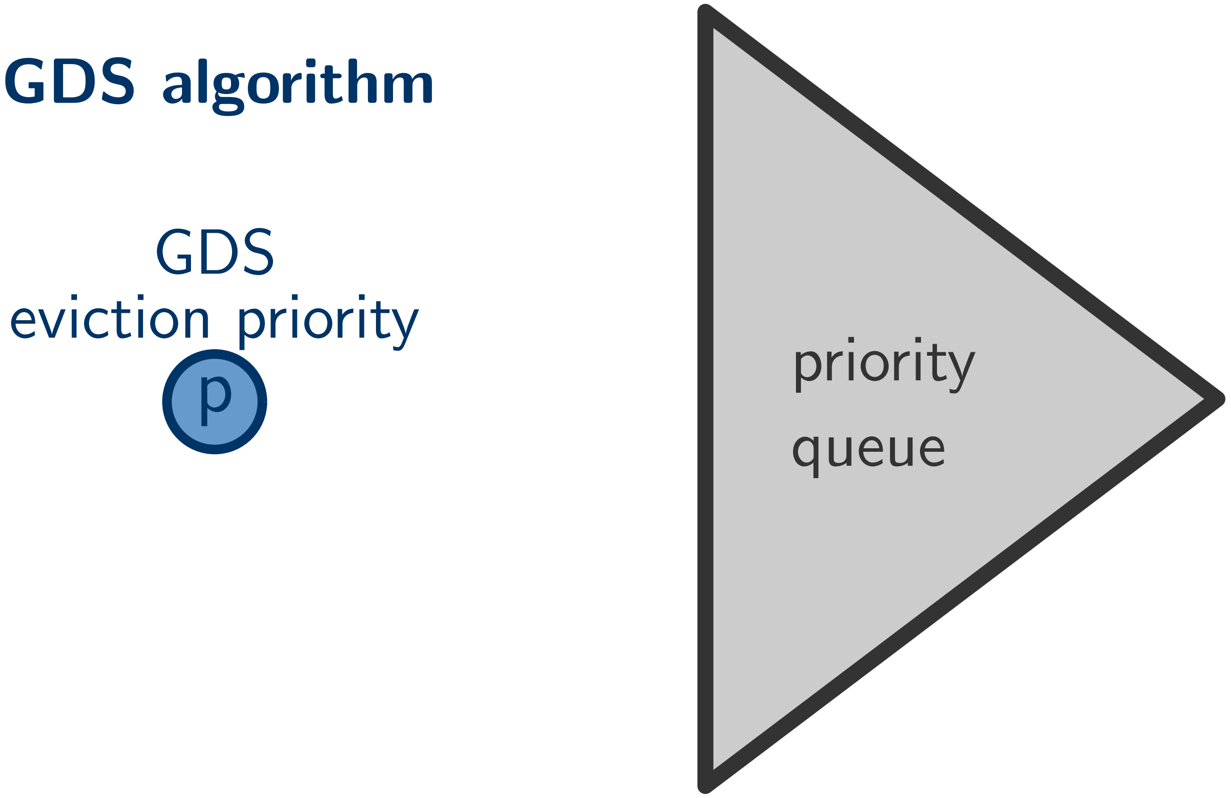
need to take **recomputation cost** into consideration

the solution

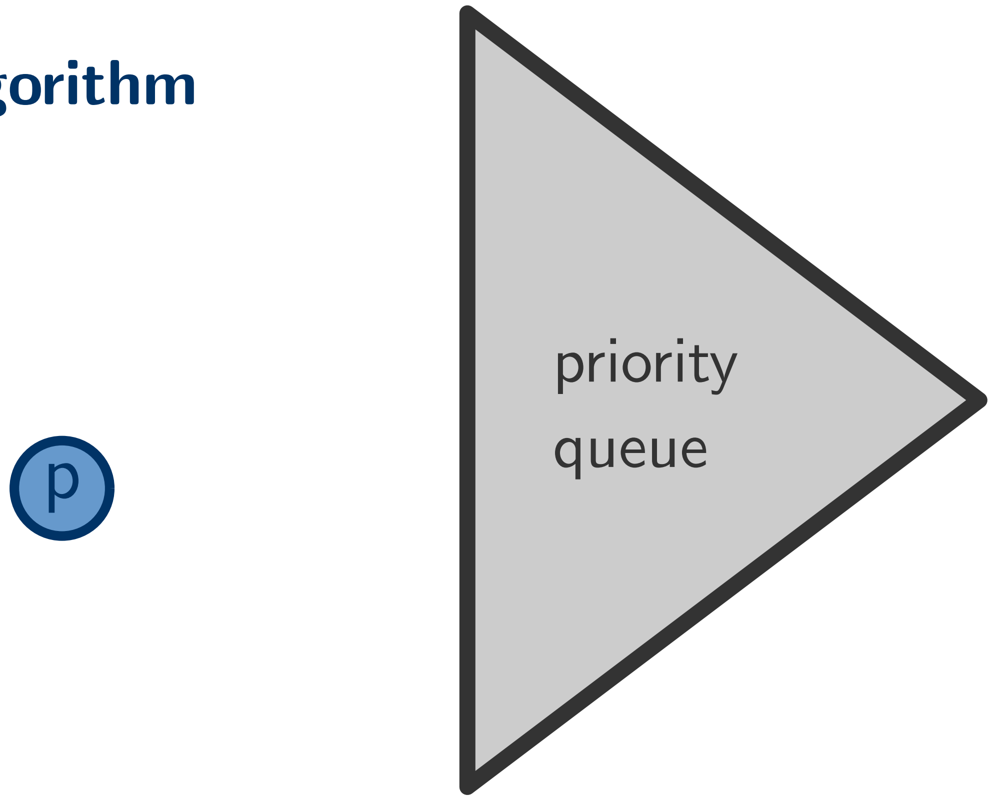

recency
recomputation cost
$
(cost/byte)
GDS eviction priority
p

GDS
eviction priority
p

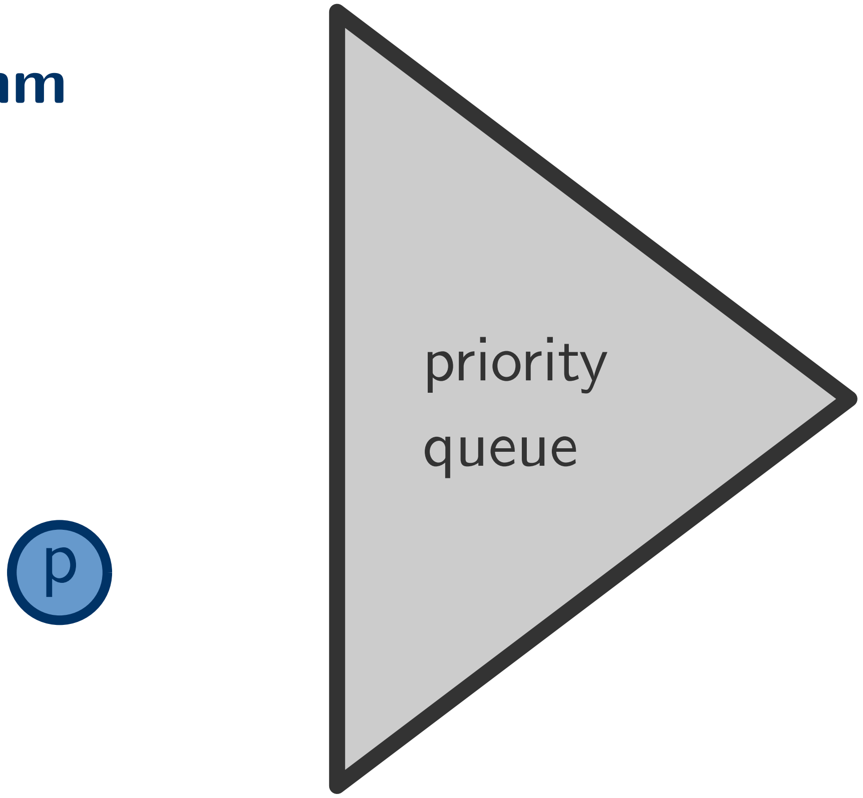
GDS algorithm



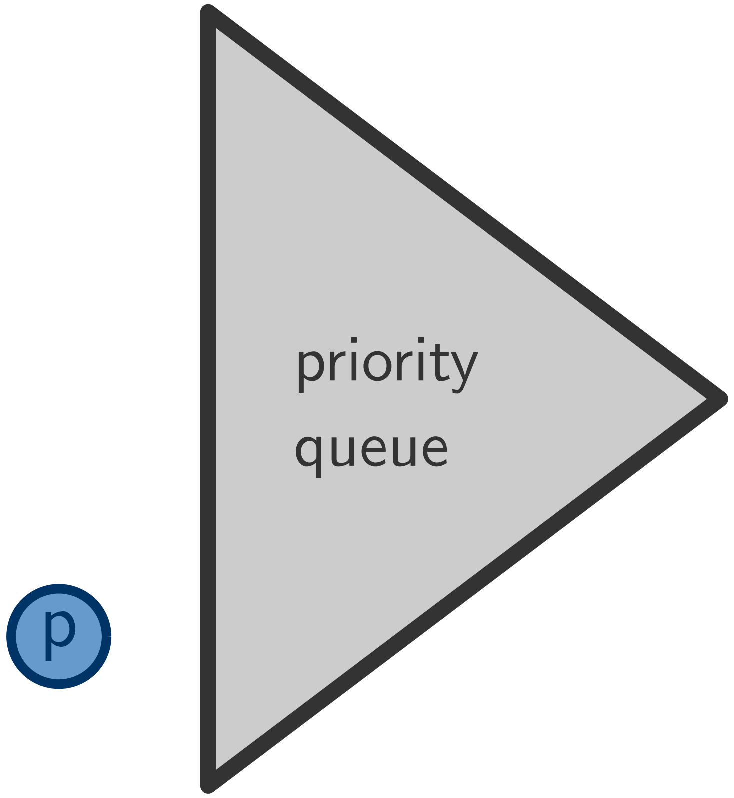
GDS algorithm



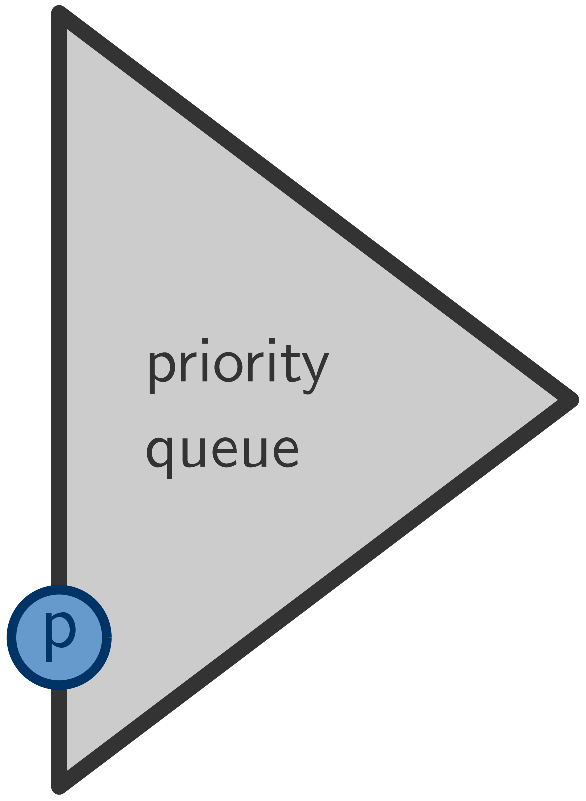
GDS algorithm



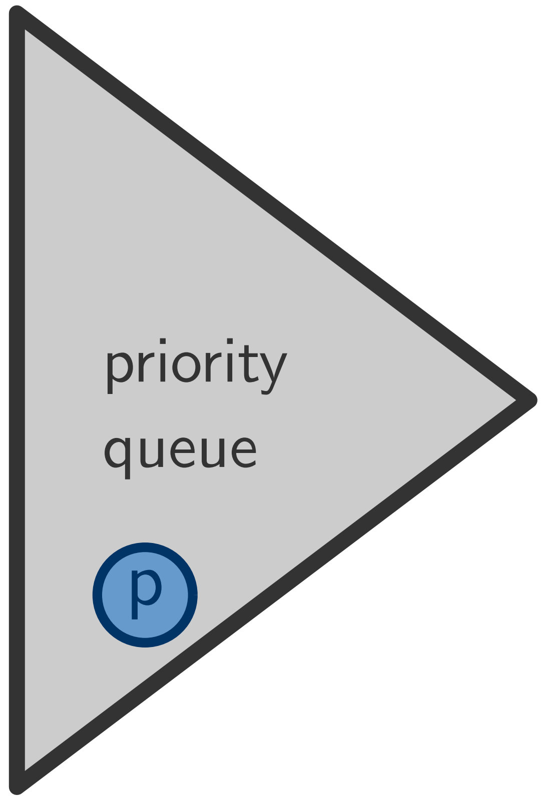
GDS algorithm



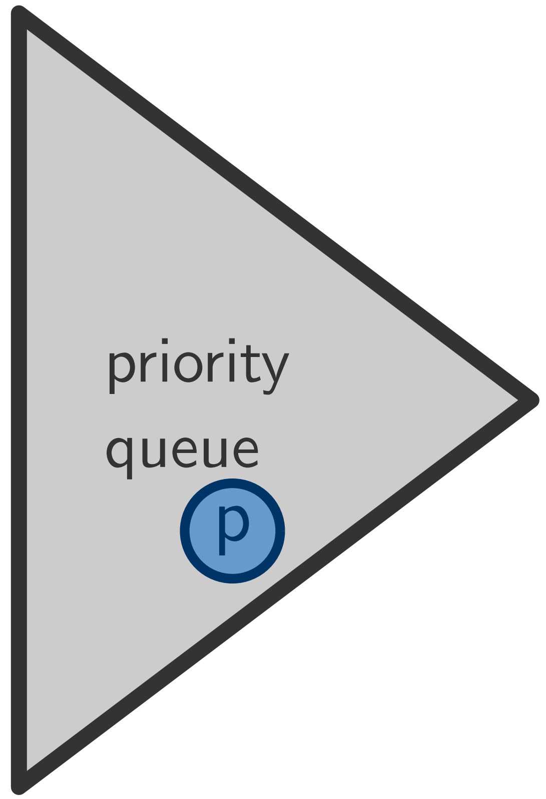
GDS algorithm



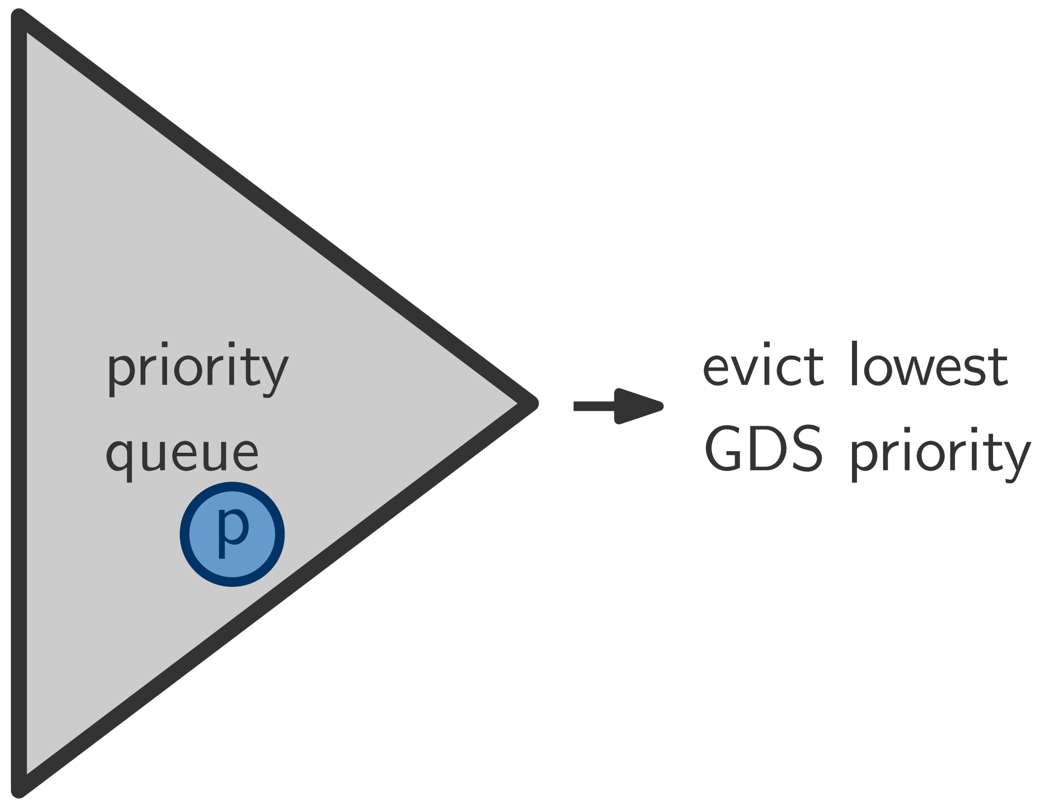
GDS algorithm



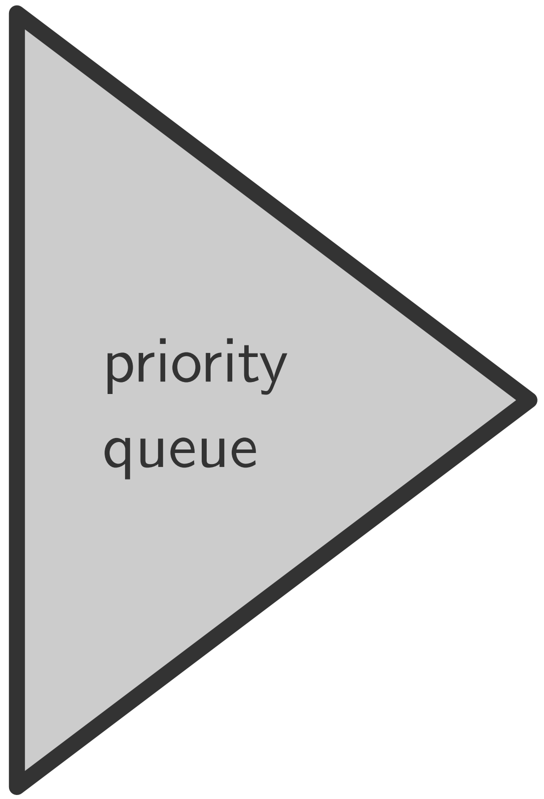
GDS algorithm



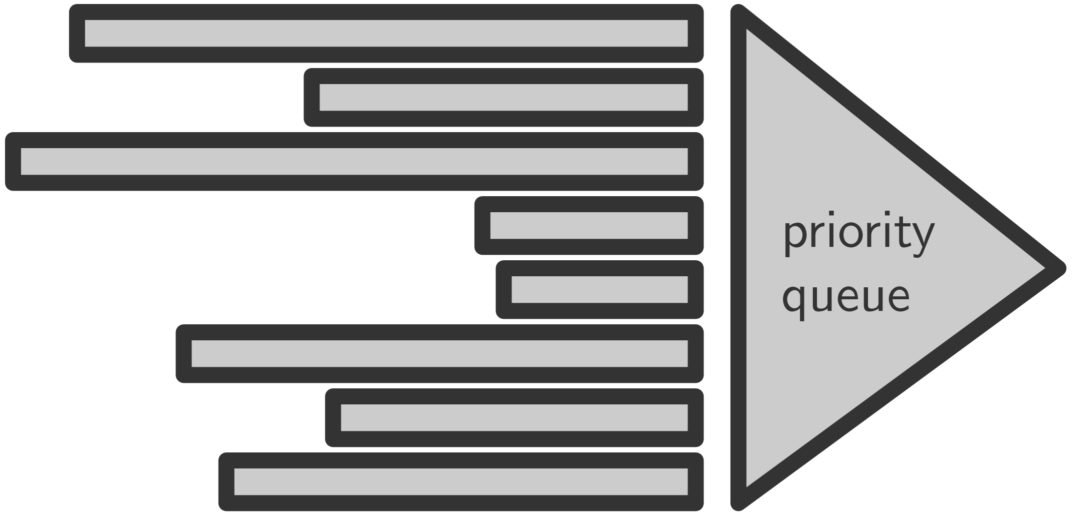
GDS algorithm



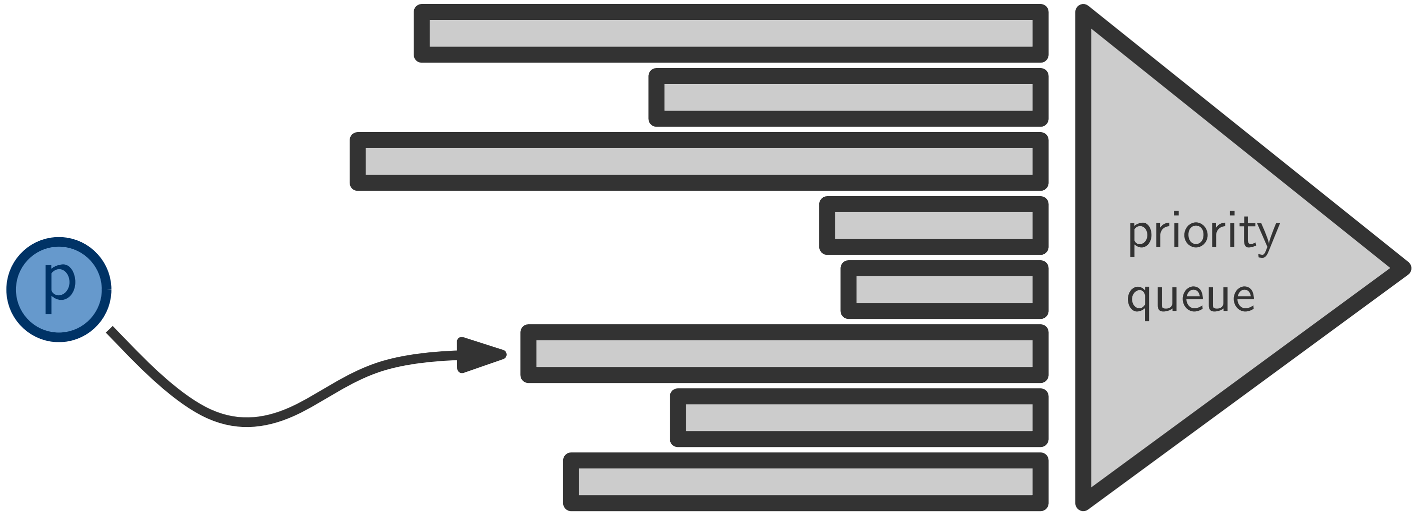
GDS algorithm



CAMP algorithm

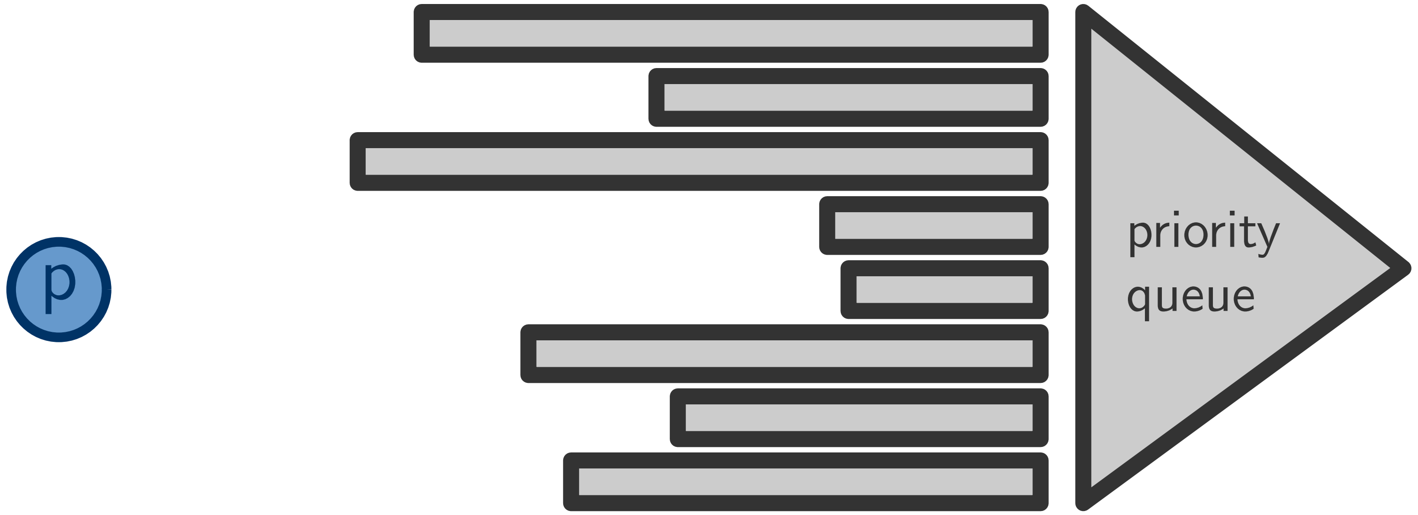


CAMP algorithm

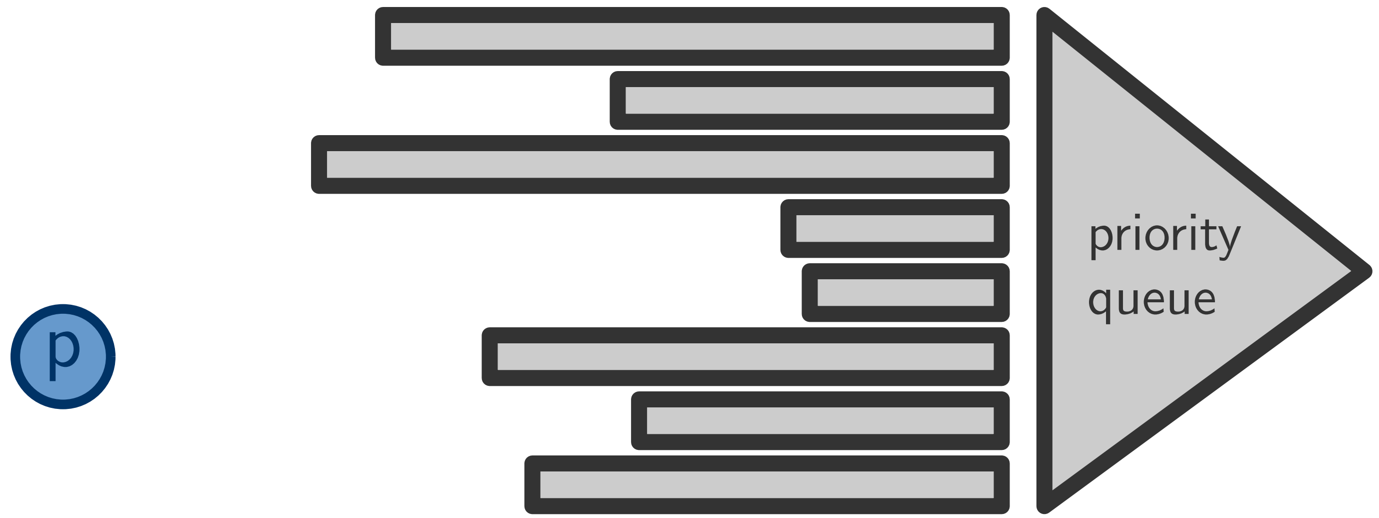


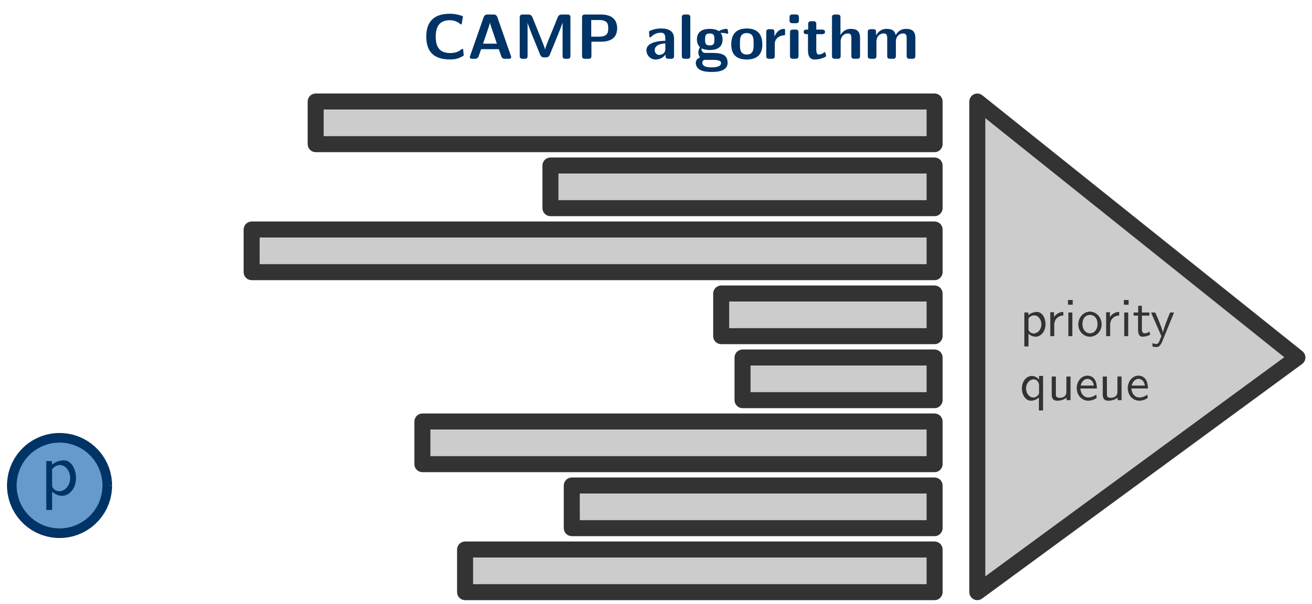
choose LRU queue
based on cost

CAMP algorithm

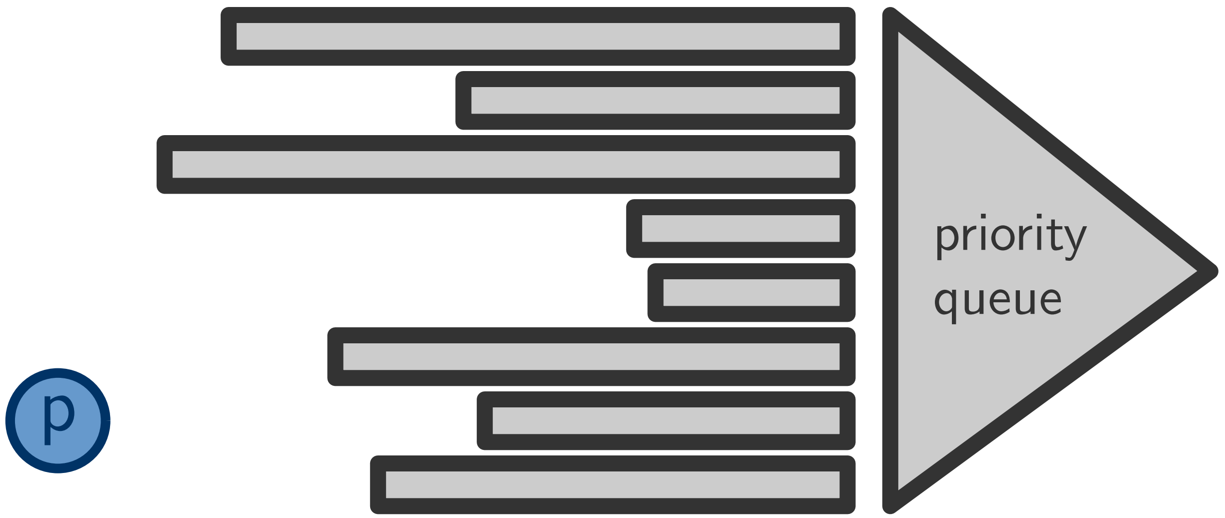


CAMP algorithm

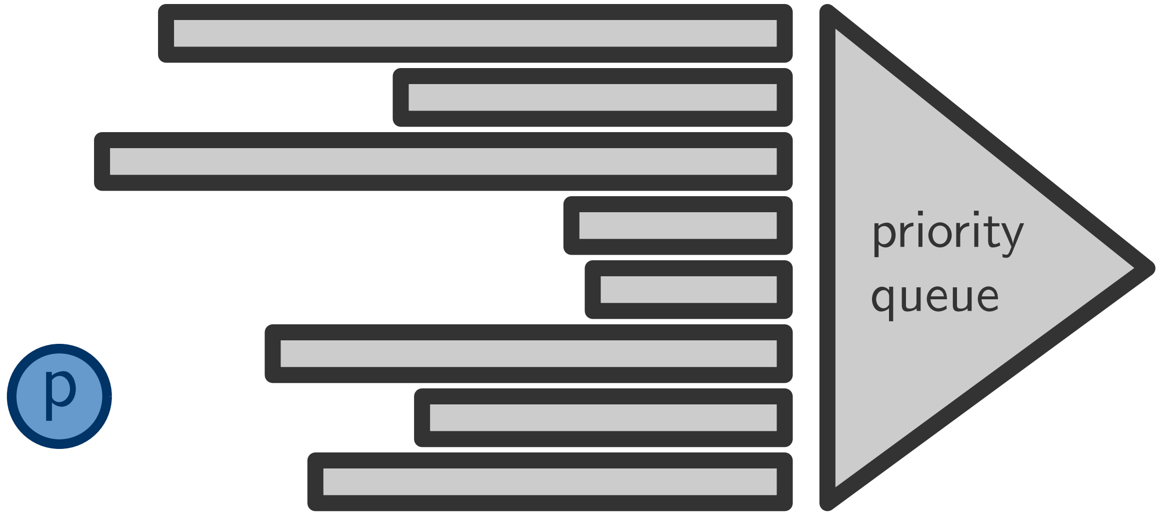



CAMP algorithm
priority
queue
p

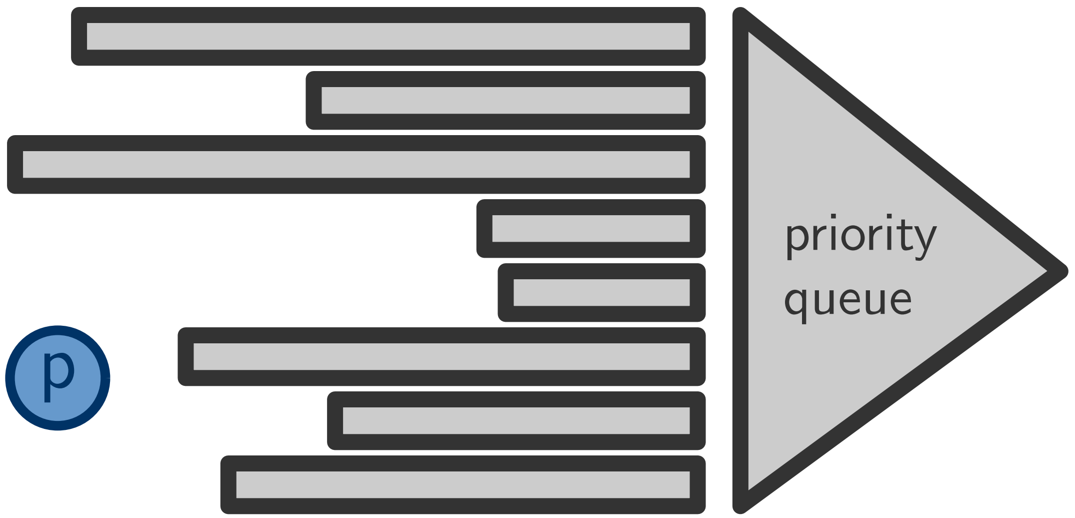
CAMP algorithm



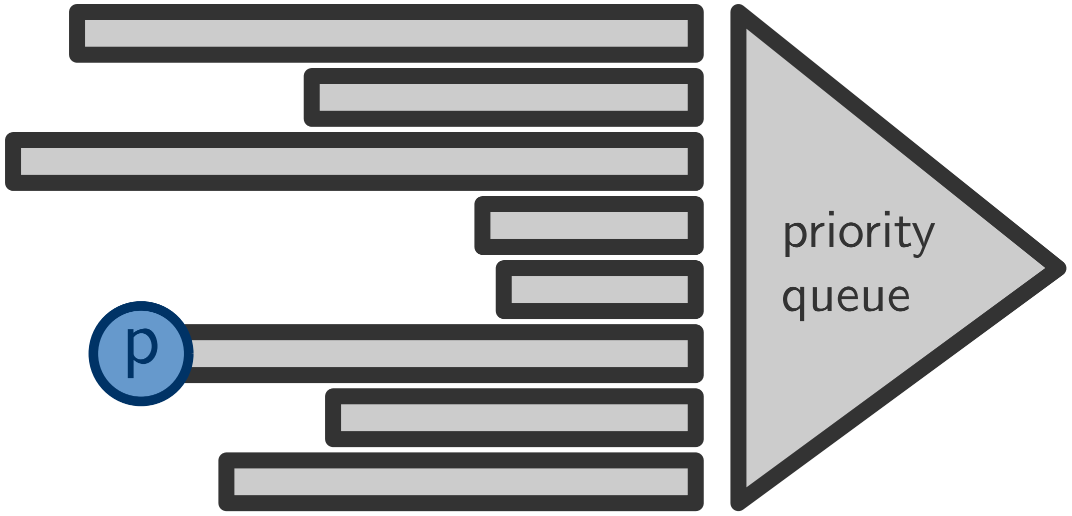
CAMP algorithm



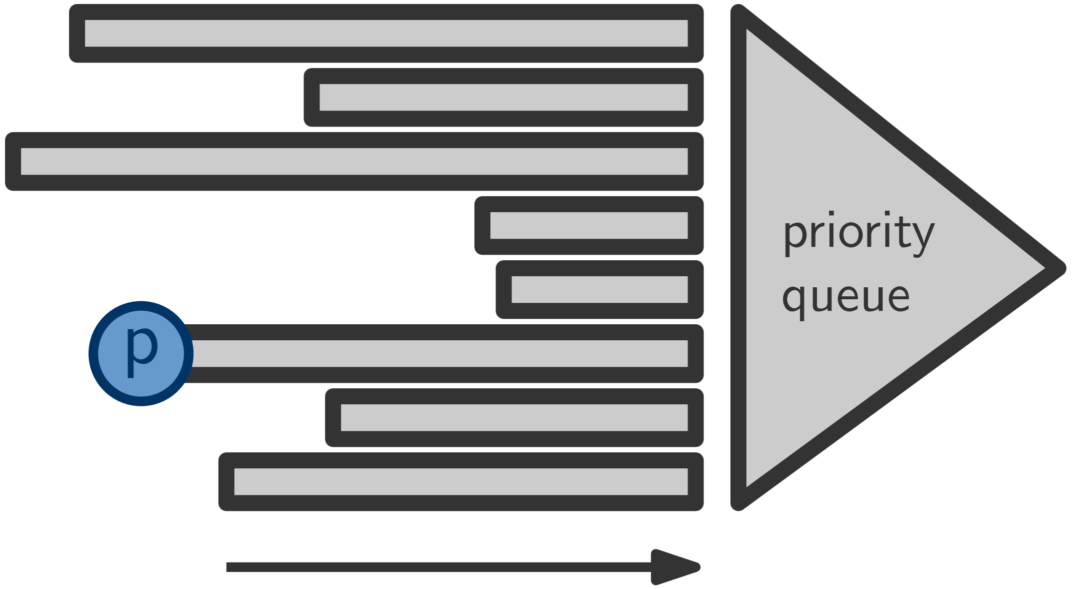
CAMP algorithm



CAMP algorithm

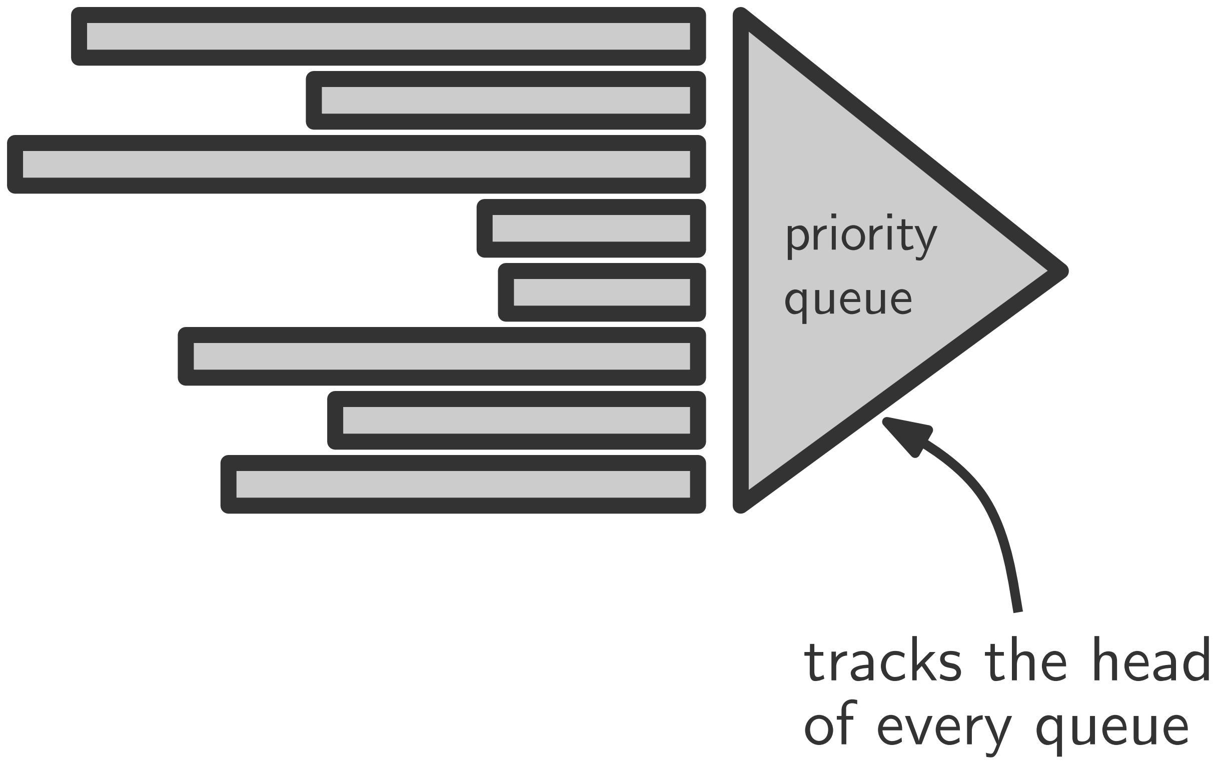


CAMP algorithm

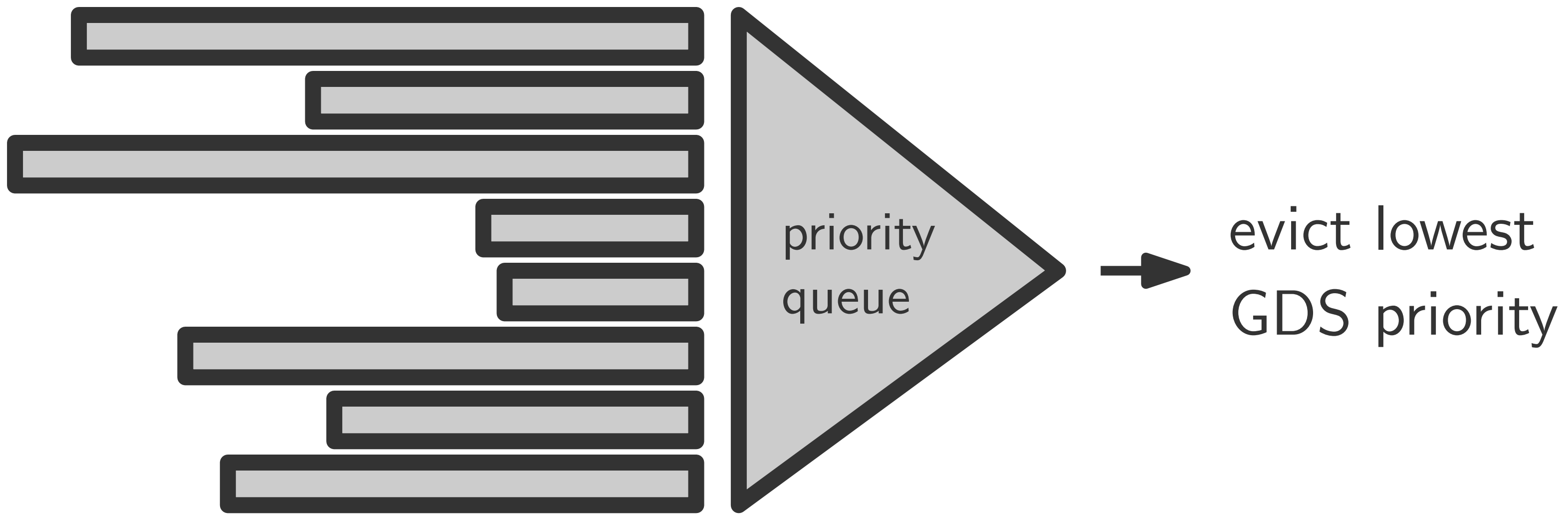


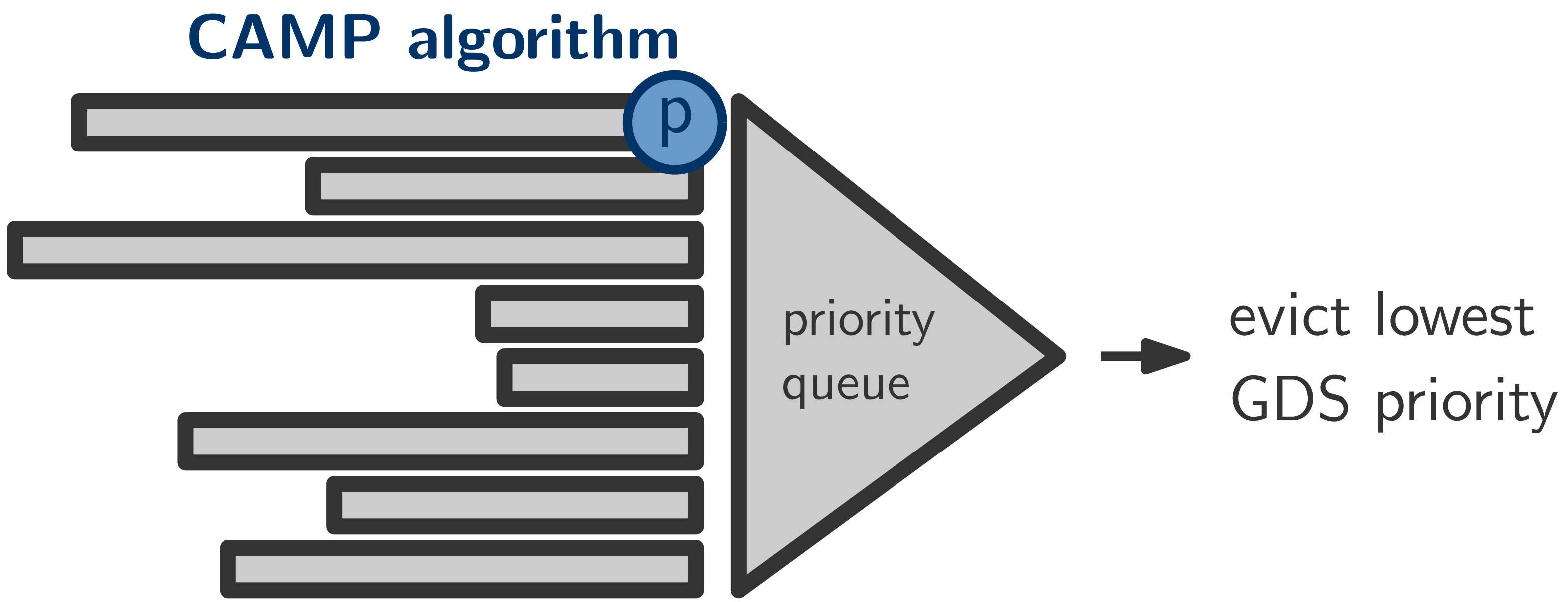
each LRU queue is sorted by GDS priority

CAMP algorithm

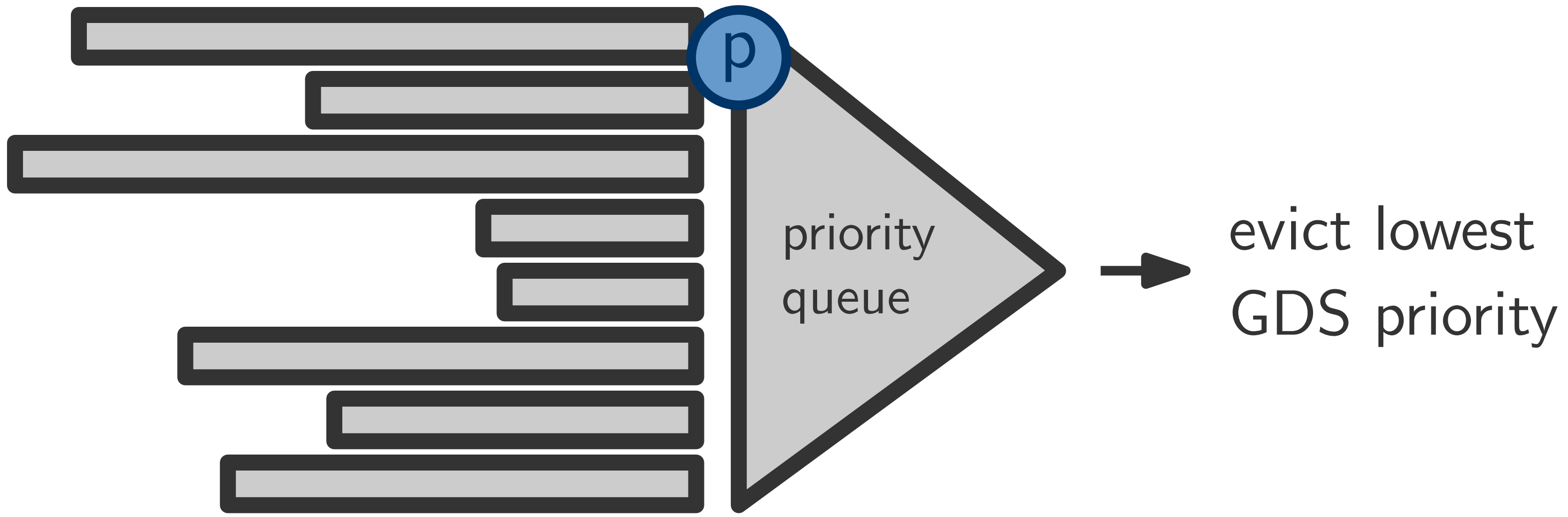


CAMP algorithm

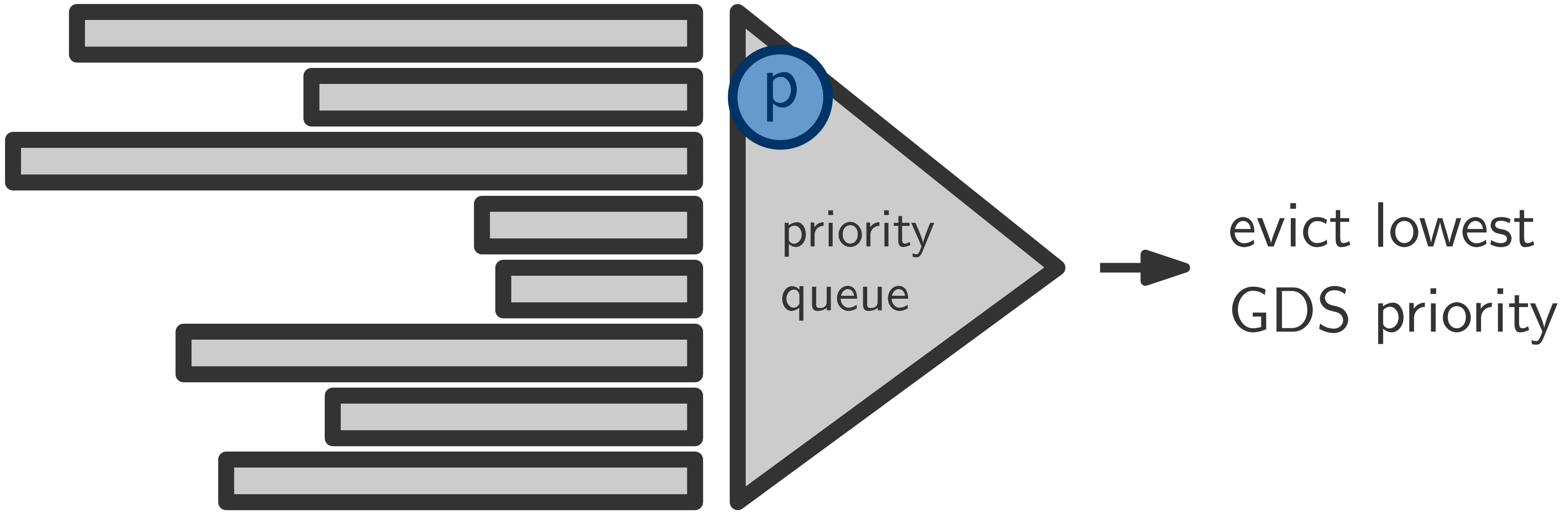



CAMP algorithm
p
priority queue
evict lowest GDS priority

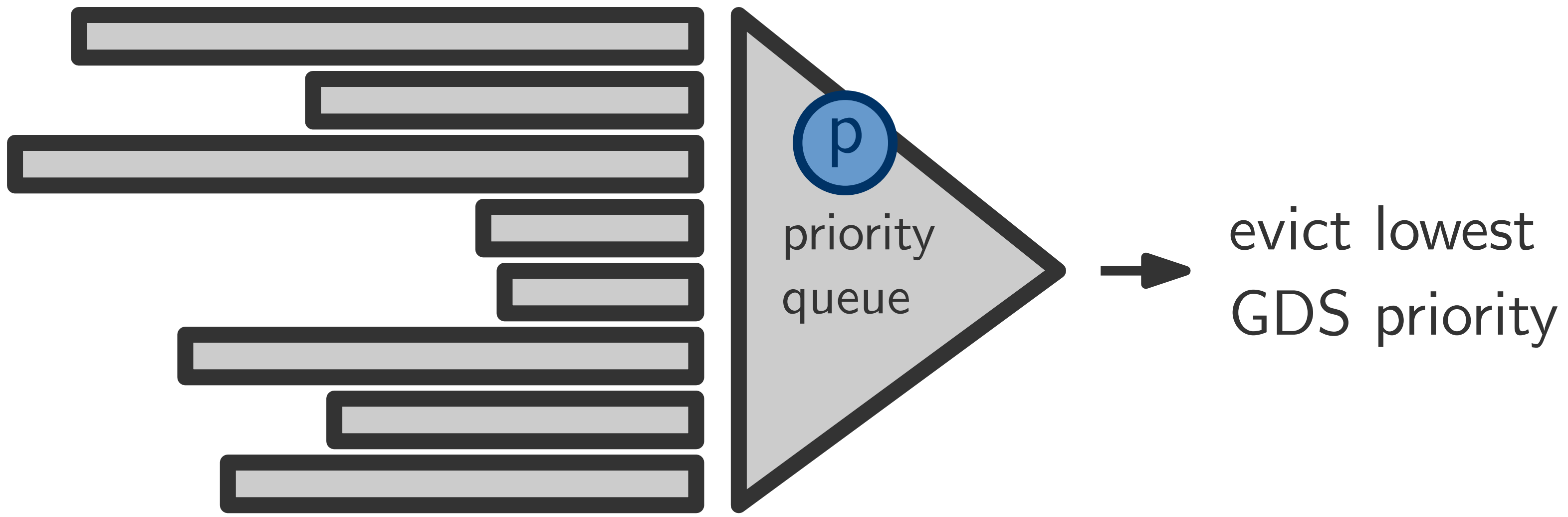
CAMP algorithm

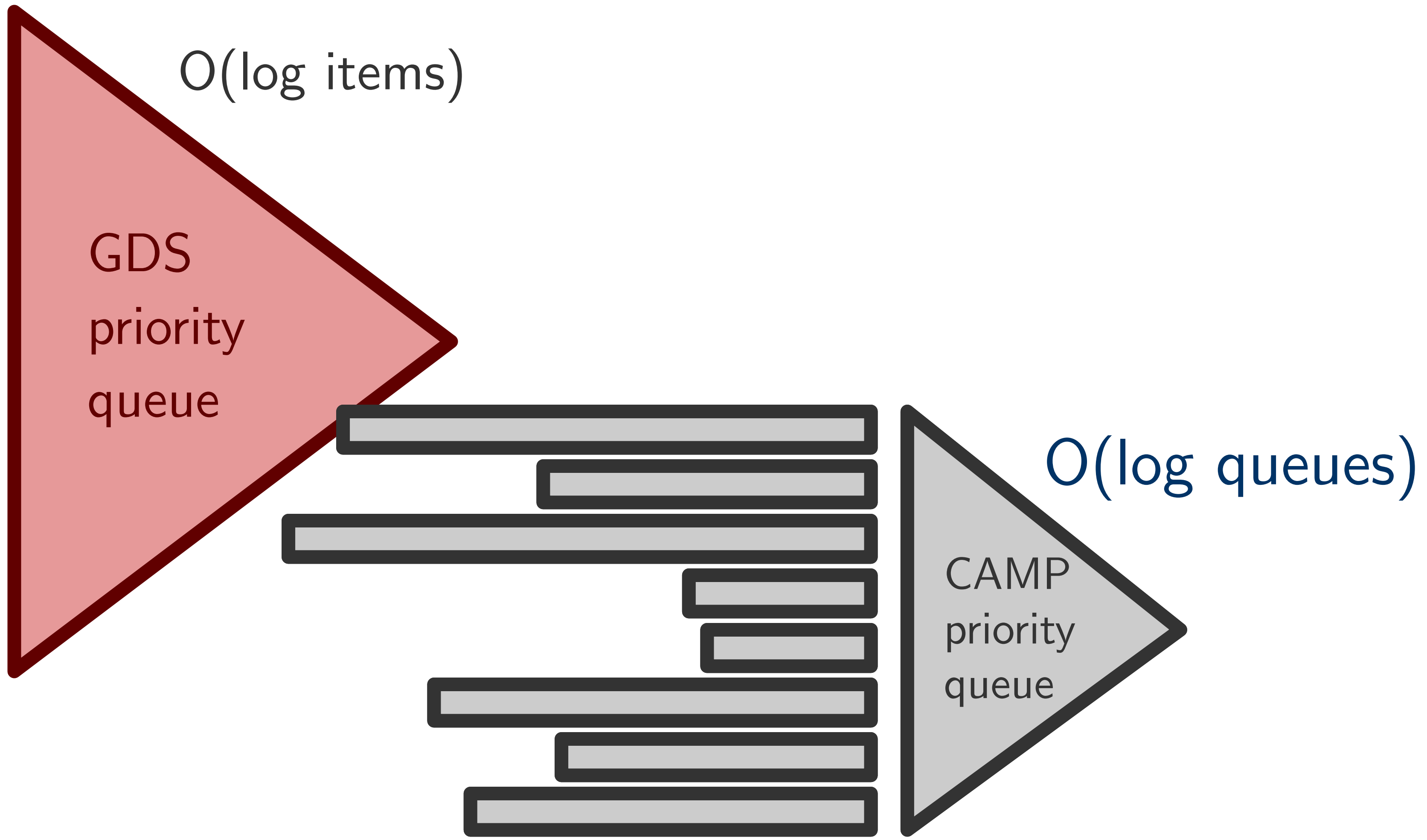


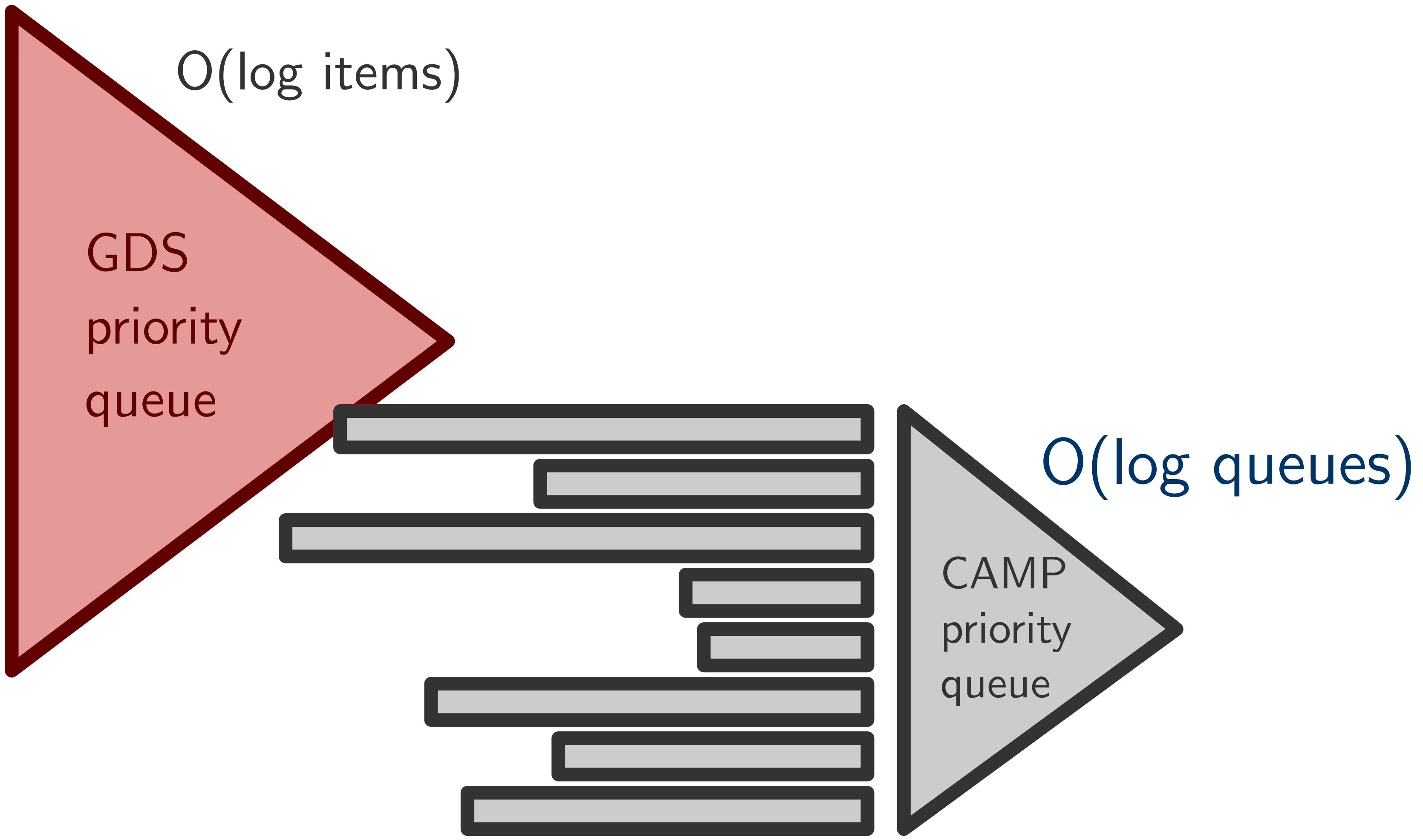
CAMP algorithm



CAMP algorithm



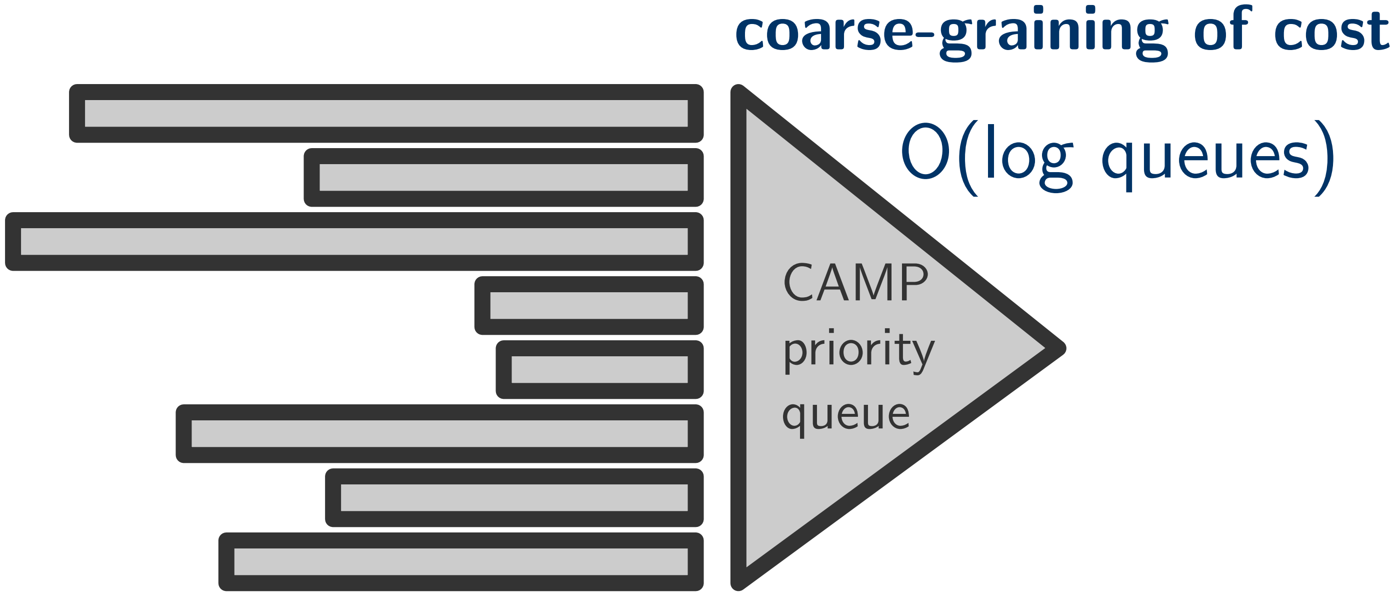

O(log items)
GDS priority queue
O(log queues)
CAMP priority queue

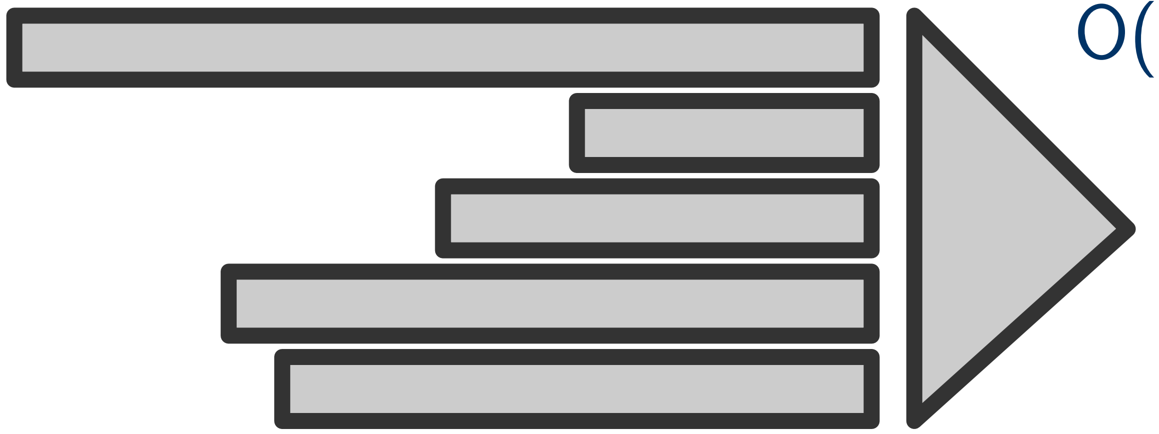


no manual intervention

no migration necessary

coarse-graining of cost





coarse-graining of cost

O(log queues)

coarse-graining of cost

O(log queues)

performance guarantee on GDS

$$\text{cost(GDS)} \leq \text{k cost(OPT)}$$

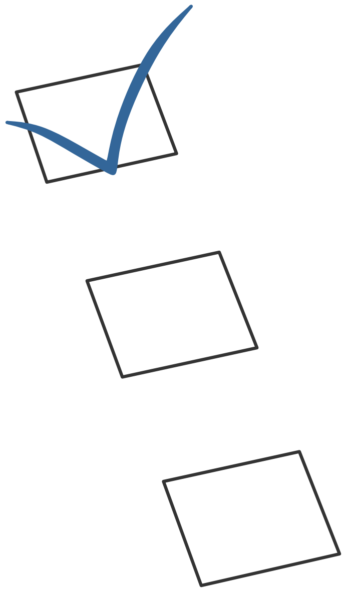
coarse-graining of cost

performance guarantee on CAMP

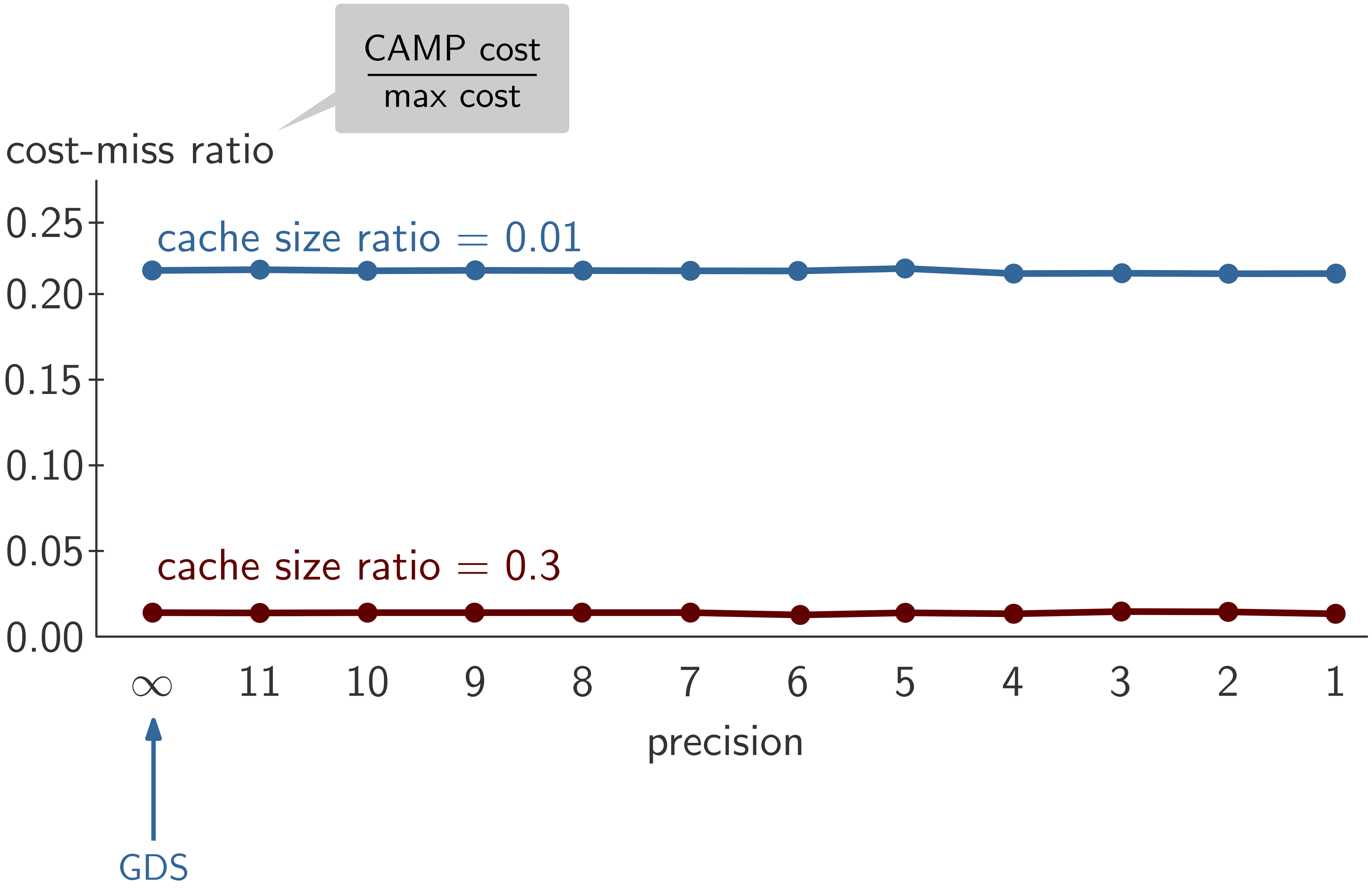
$$\text{cost(CAMP)} \leq (1 + \varepsilon)\text{k cost(OPT)}$$

approximation parameter

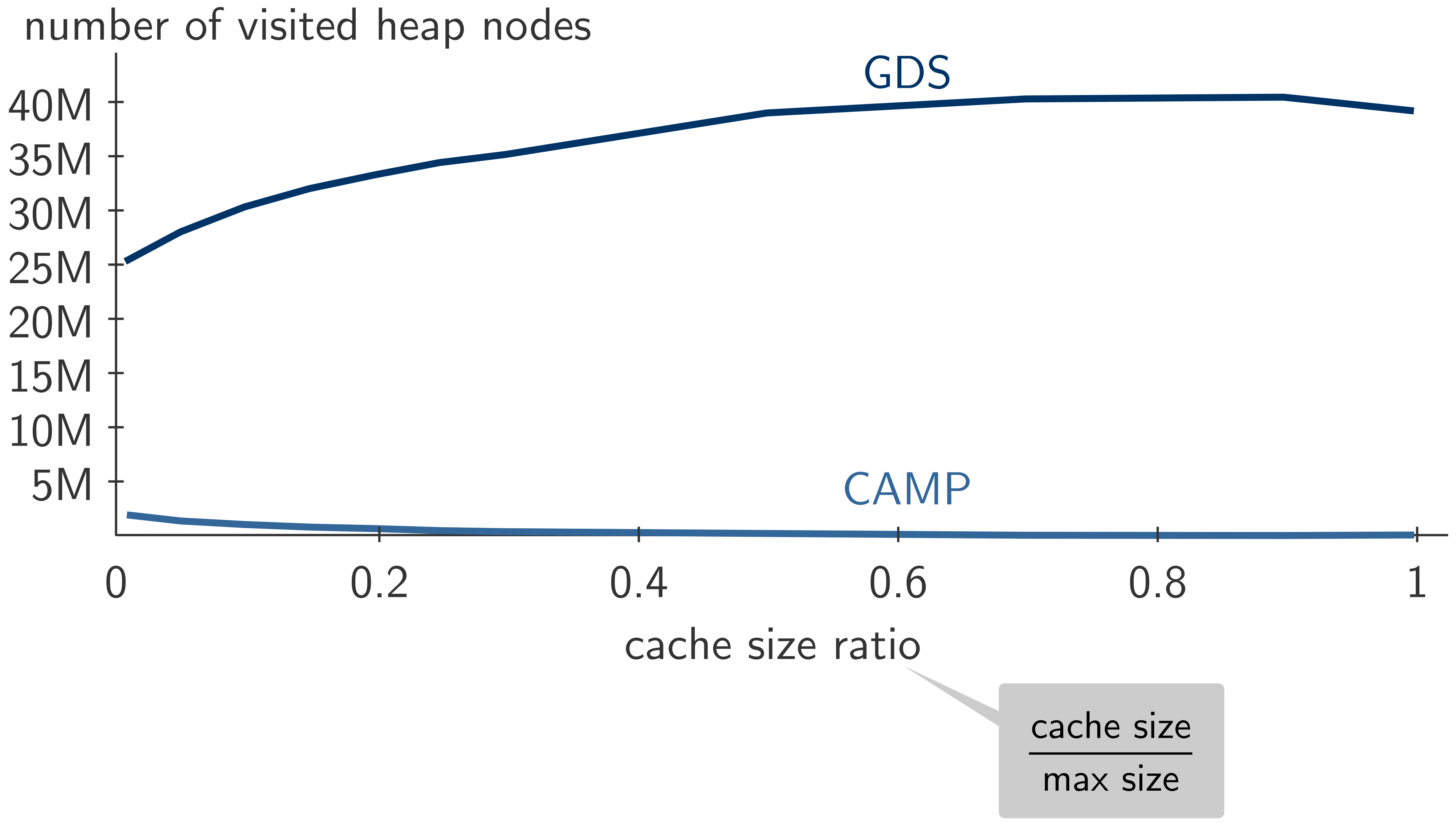
the evaluation



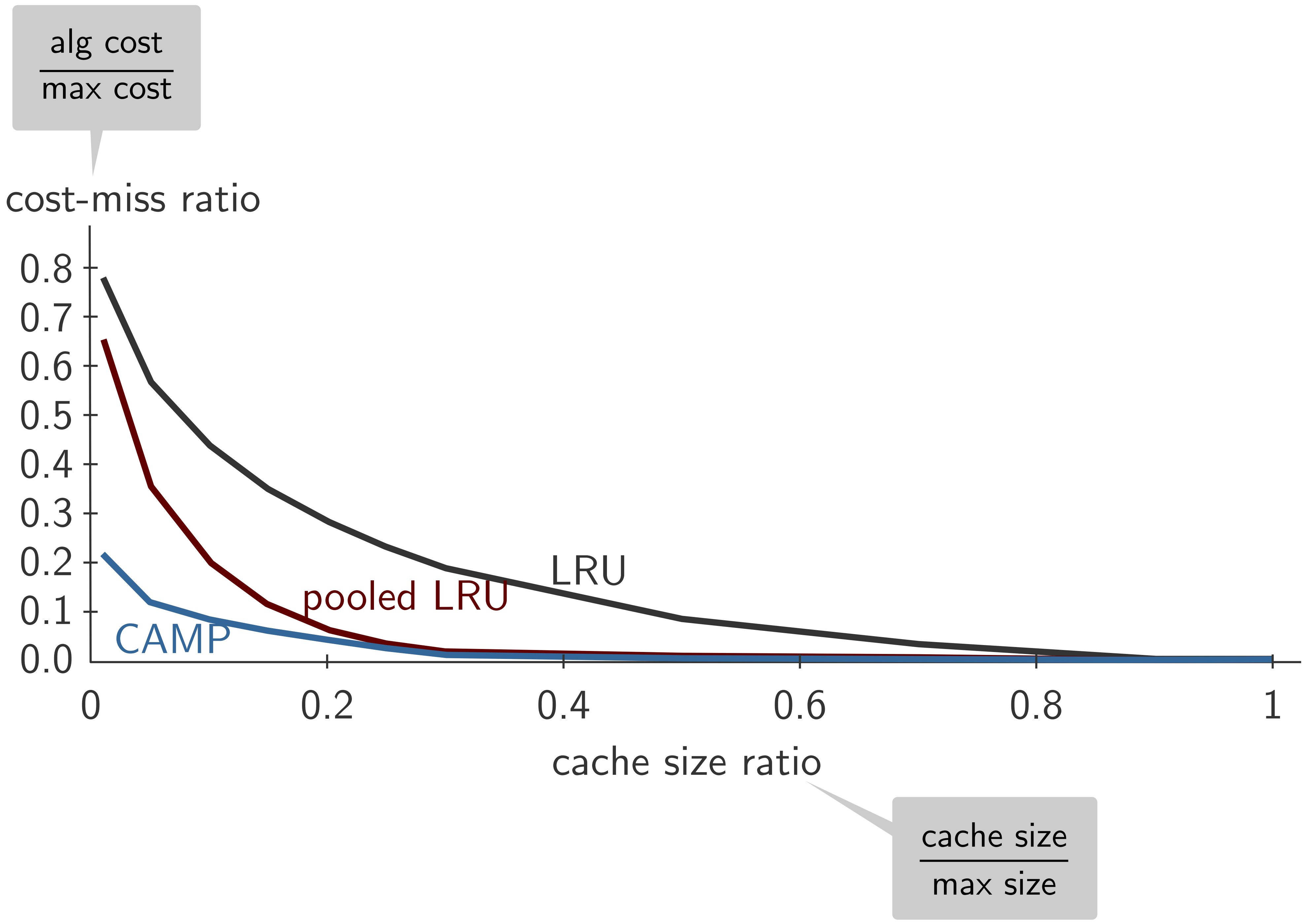
CAMP is resilient to approximation



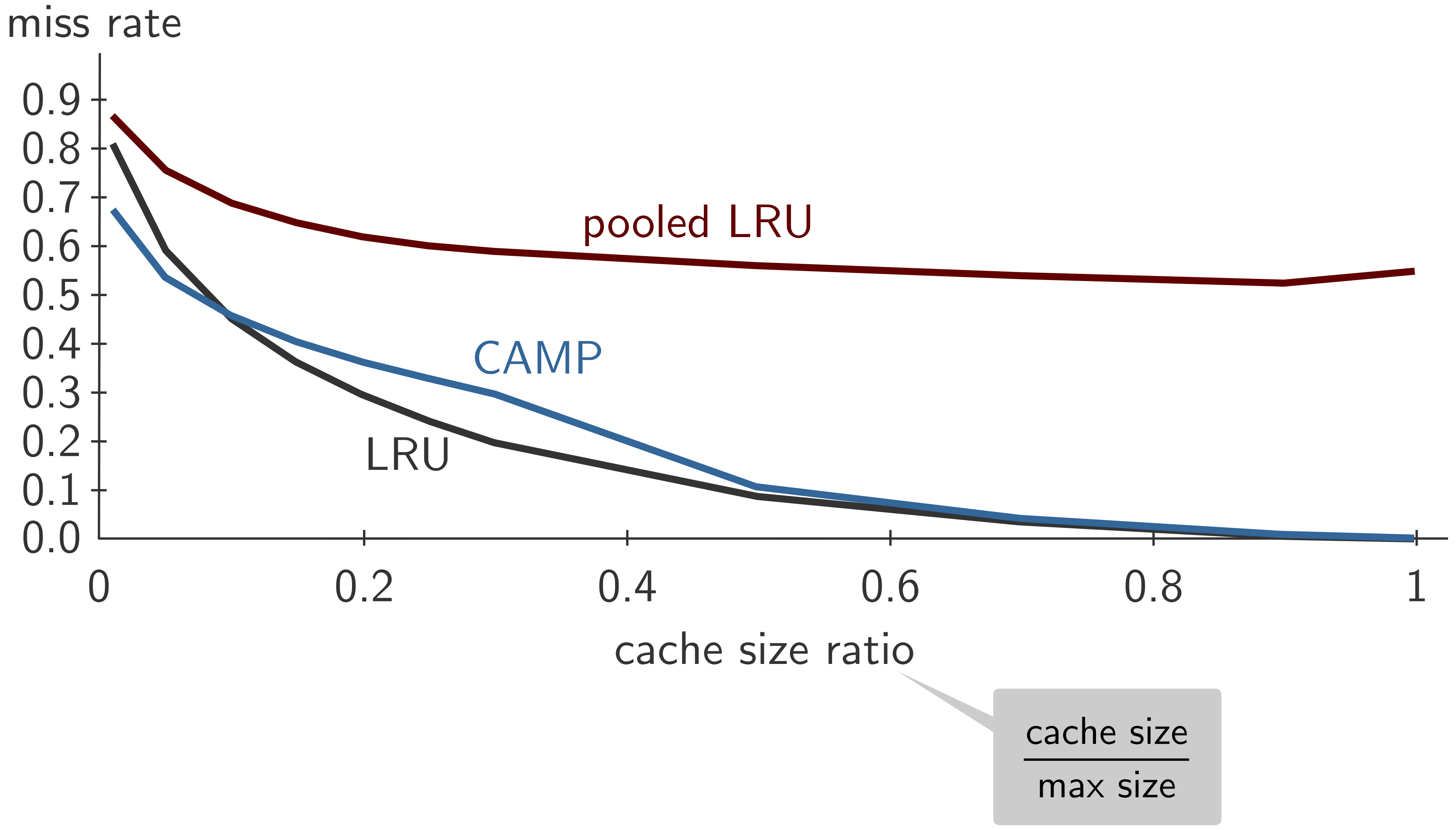
CAMP incurs little overhead



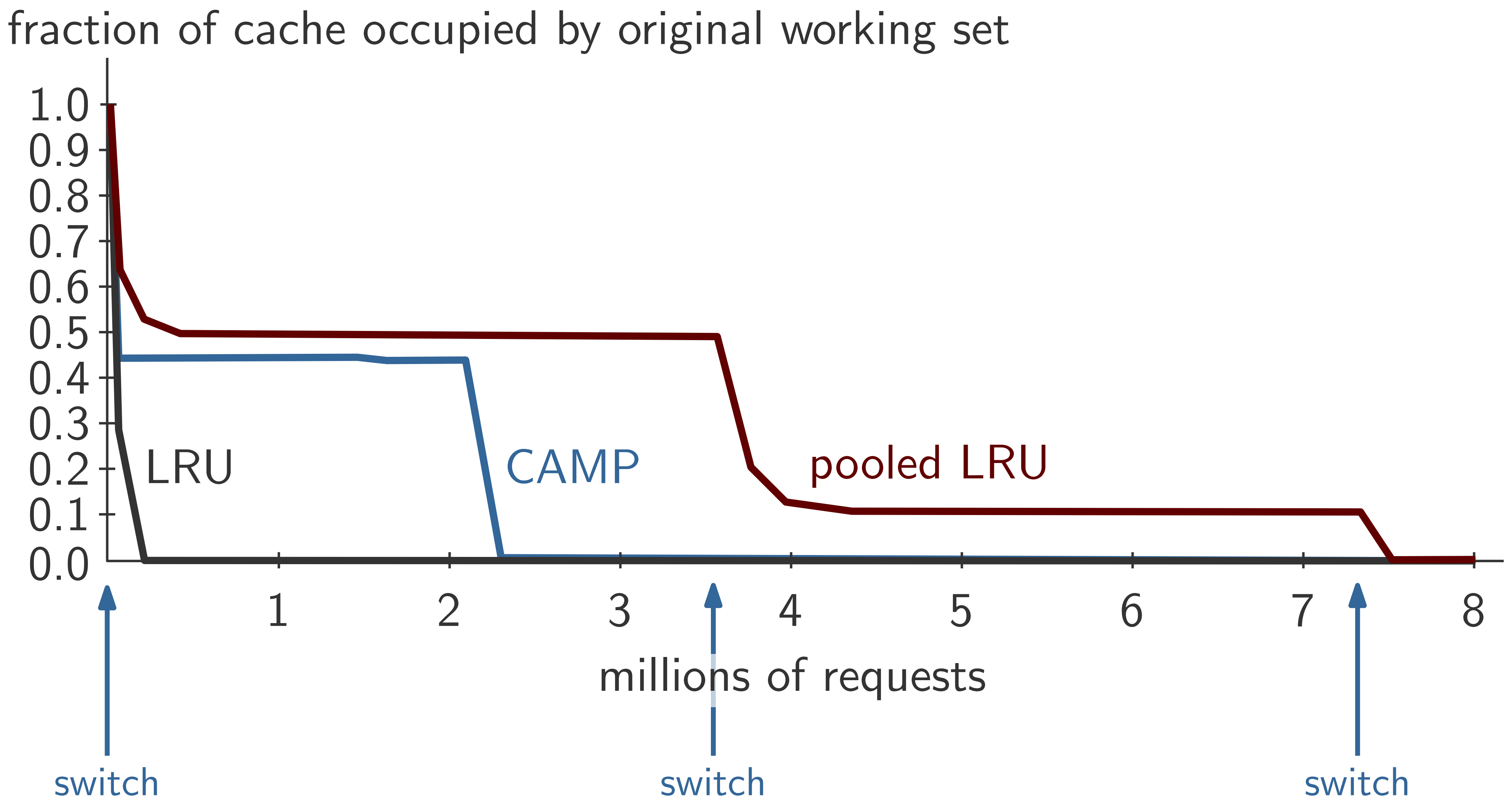
CAMP beats LRU and pooled LRU

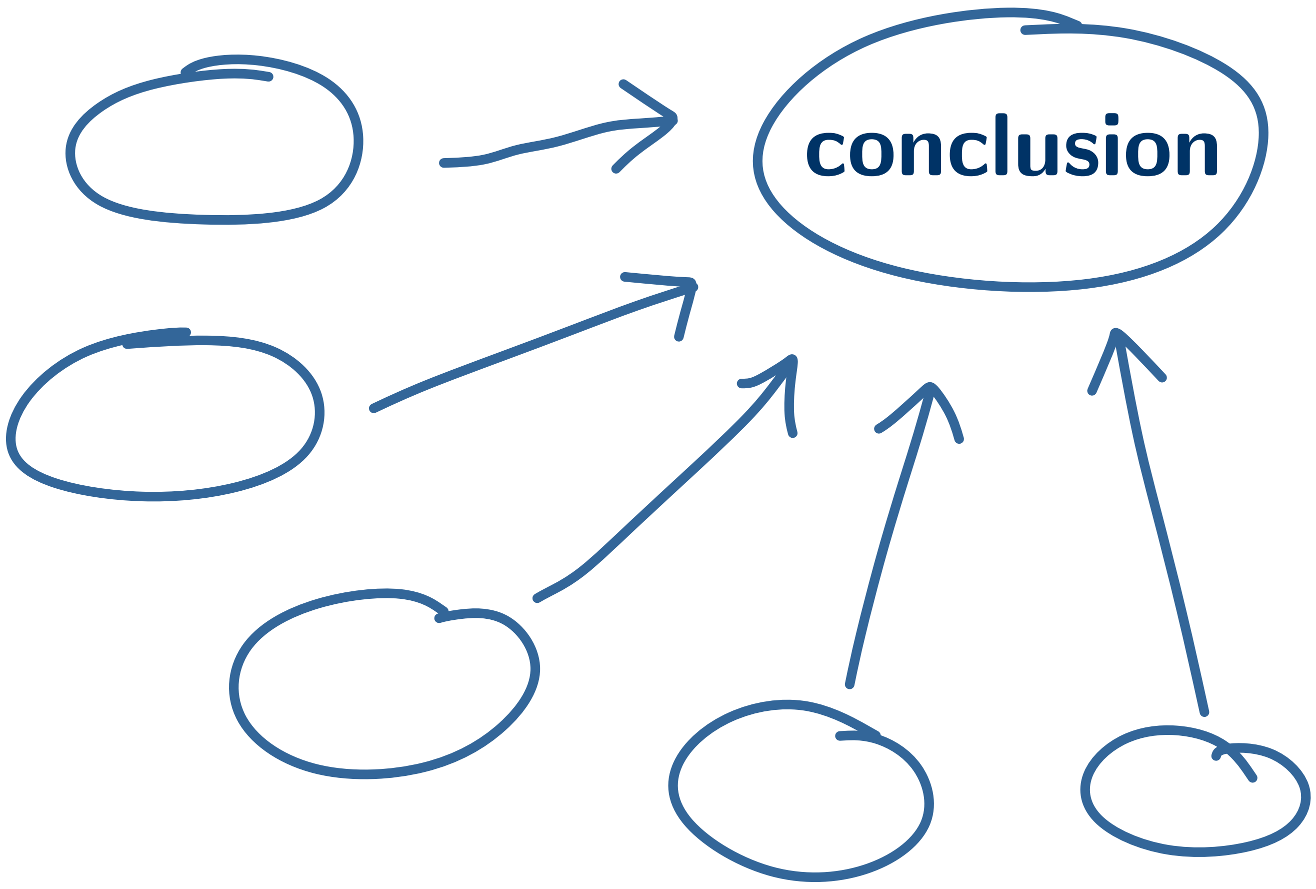


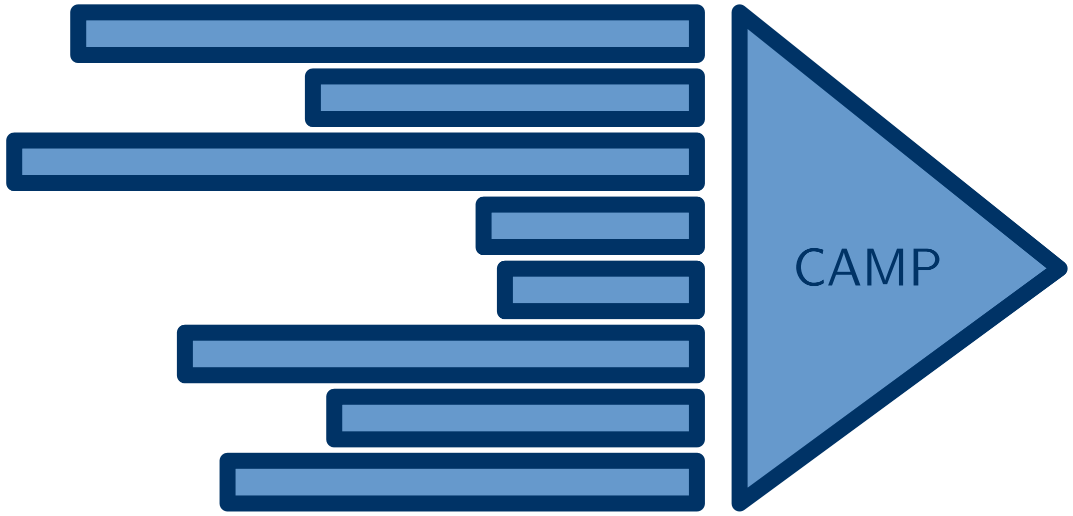
CAMP beats LRU and pooled LRU



CAMP handles churn gracefully




conclusion



- ✓ performs as well as GDS without the overhead
- ✓ performs better than LRU and pooled LRU
- ✓ self-tuning
- ✓ handles evolving access pattern

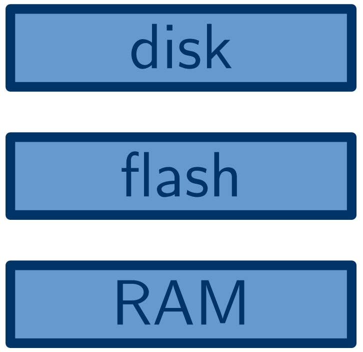
Future research directions

design features

management of memory allocation
Cache replacement with memory allocation, ALENEX 2015

admission control

applications



caching in a hierarchical tiered storage system

cooperative caching framework