Harvesting Idle Memory for Application-Managed Soft State with Midas

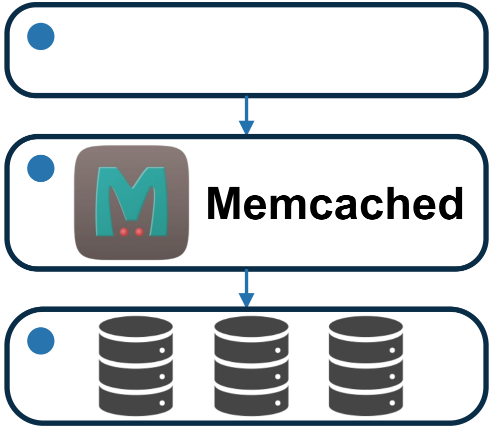
Yifan Qiao, Zhenyuan Ruan, Haoran Ma

Adam Belay, Miryung Kim, Harry Xu

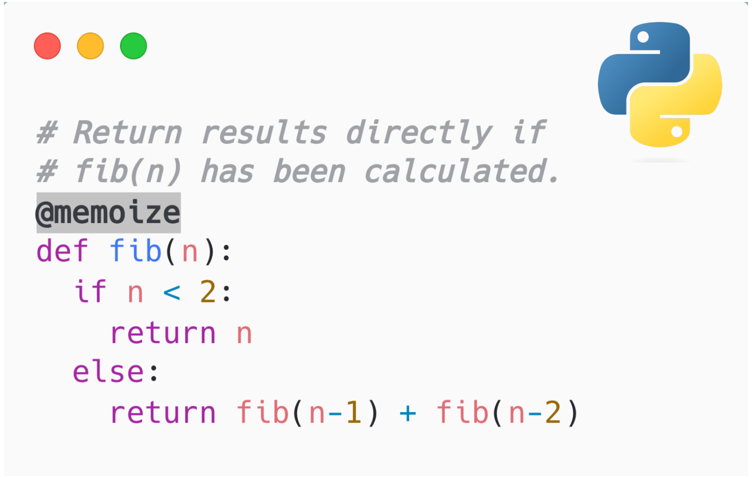
Soft State Is Everywhere

Increases performance but safe to discard

Examples:



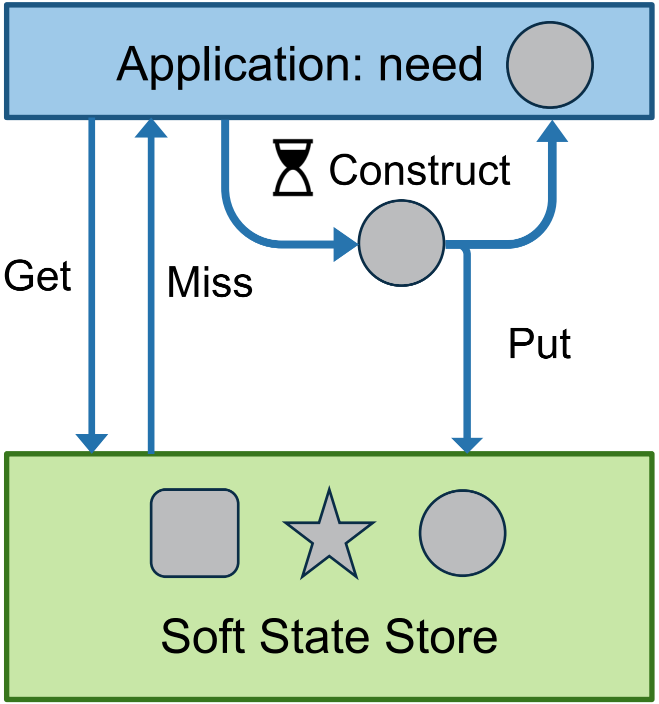
Cache



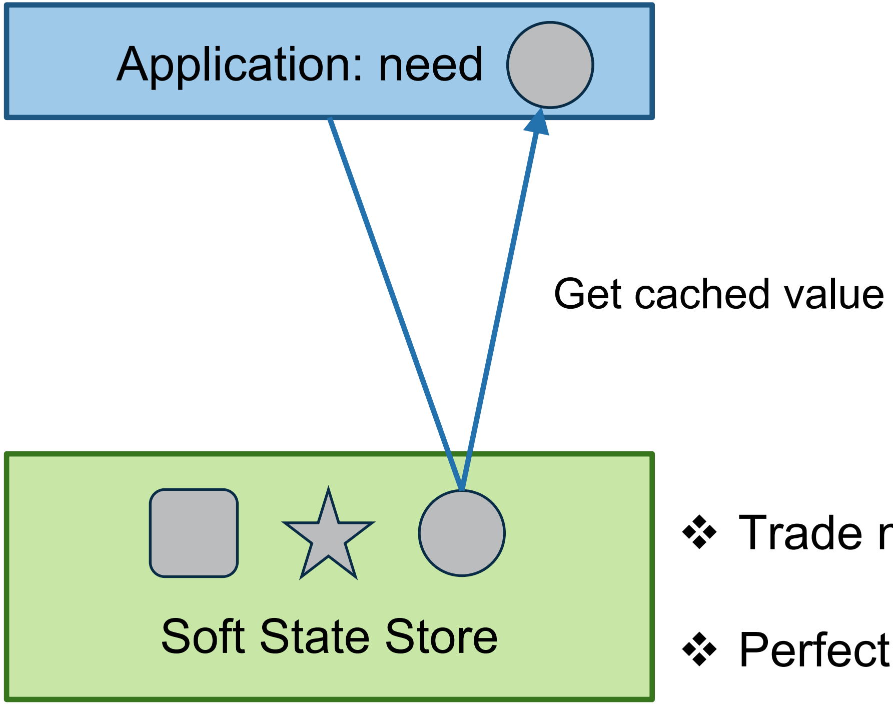
```
# Return results directly if
# fib(n) has been calculated.
@memoize
def fib(n):
  if n < 2:
    return n
  else:
    return fib(n-1) + fib(n-2)
```

Memoization

Soft State Is Everywhere

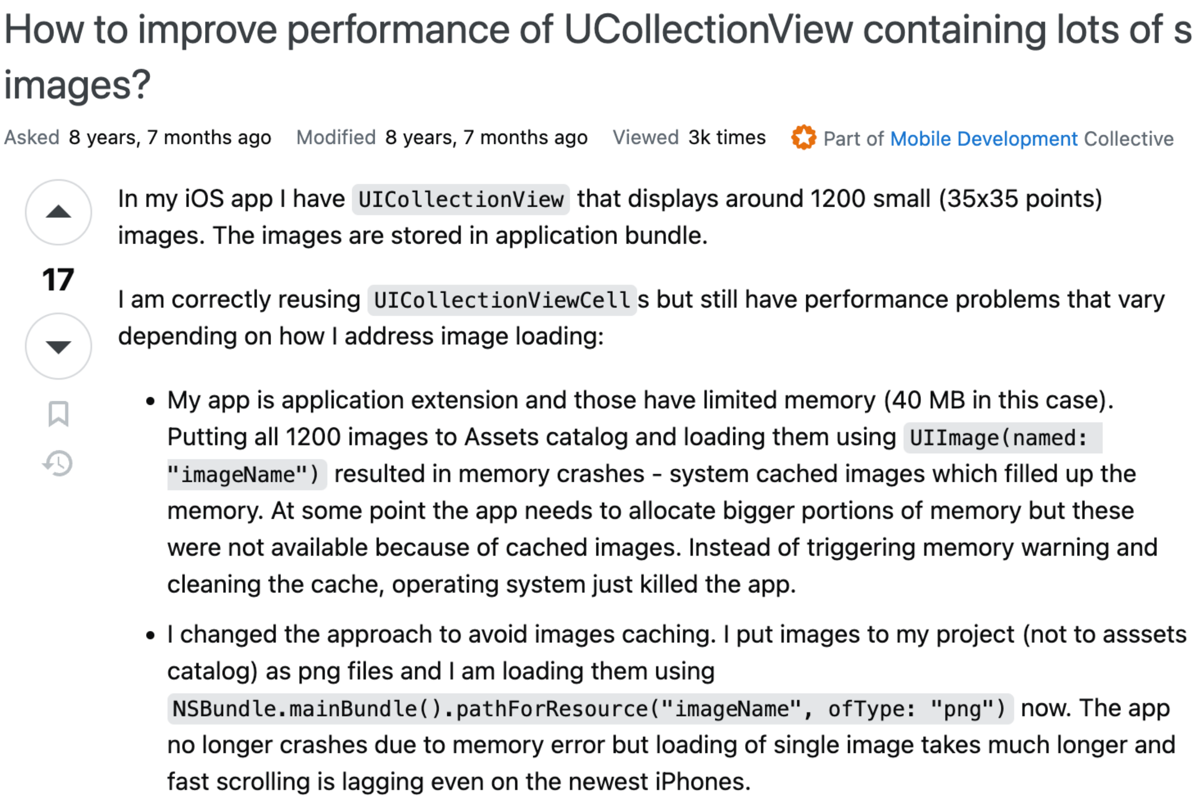


Soft State Is Everywhere



- Trade memory for performance
- Perfect use case for idle memory

Managing Soft State Is Hard



How to improve performance of UCollectionView containing lots of small images?

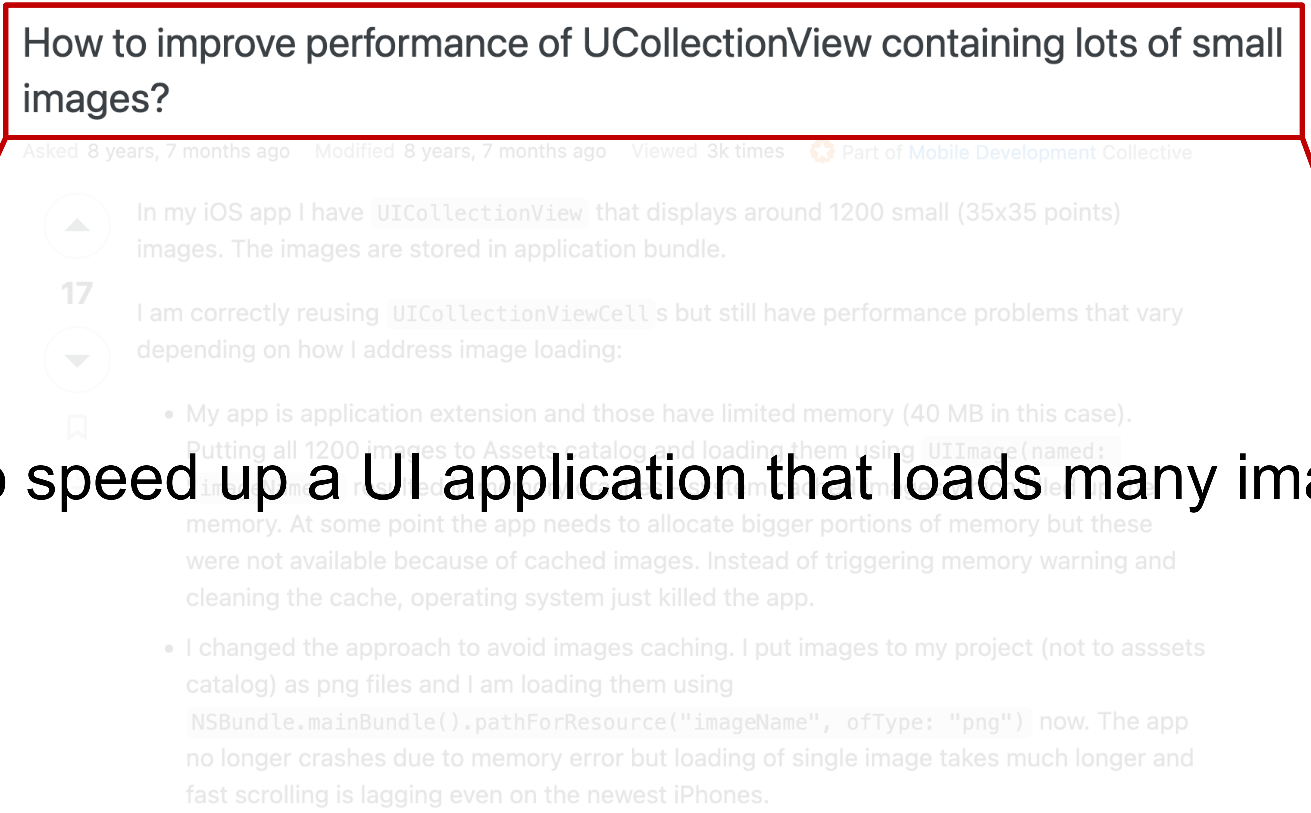
Asked 8 years, 7 months ago Modified 8 years, 7 months ago Viewed 3k times Part of Mobile Development Collective

In my iOS app I have `UICollectionView` that displays around 1200 small (35x35 points) images. The images are stored in application bundle.

I am correctly reusing `UICollectionViewCell`s but still have performance problems that vary depending on how I address image loading:

- My app is application extension and those have limited memory (40 MB in this case). Putting all 1200 images to Assets catalog and loading them using `UIImage(named: "imageName")` resulted in memory crashes - system cached images which filled up the memory. At some point the app needs to allocate bigger portions of memory but these were not available because of cached images. Instead of triggering memory warning and cleaning the cache, operating system just killed the app.
- I changed the approach to avoid images caching. I put images to my project (not to asssets catalog) as png files and I am loading them using `NSBundle.mainBundle().pathForResource("imageName", ofType: "png")` now. The app no longer crashes due to memory error but loading of single image takes much longer and fast scrolling is lagging even on the newest iPhones.

Managing Soft State Is Hard



How to improve performance of UCollectionView containing lots of small images?

Asked 8 years, 7 months ago Modified 8 years, 7 months ago Viewed 3k times Part of Mobile Development Collective

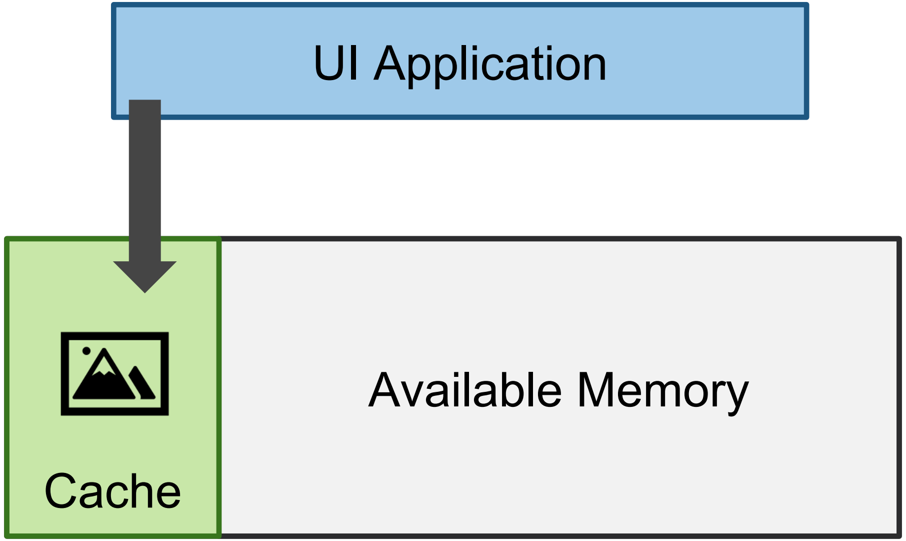
In my iOS app I have `UICollectionView` that displays around 1200 small (35x35 points) images. The images are stored in application bundle.

I am correctly reusing `UICollectionViewCell`s but still have performance problems that vary depending on how I address image loading:

- My app is application extension and those have limited memory (40 MB in this case). Putting all 1200 images to Assets catalog and loading them using `UIImage(named: imageName)` resulted in memory issues - system cached images and filled up the memory. At some point the app needs to allocate bigger portions of memory but these were not available because of cached images. Instead of triggering memory warning and cleaning the cache, operating system just killed the app.
- I changed the approach to avoid images caching. I put images to my project (not to asssets catalog) as png files and I am loading them using `NSBundle.mainBundle().pathForResource("imageName", ofType: "png")` now. The app no longer crashes due to memory error but loading of single image takes much longer and fast scrolling is lagging even on the newest iPhones.

How to speed up a UI application that loads many images?

Option 1: Storing All Soft State

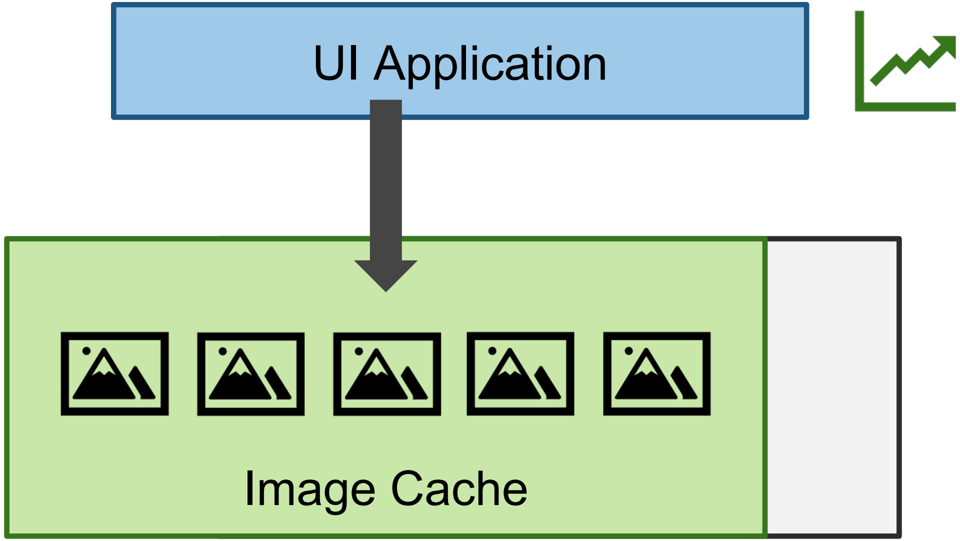


UI Application

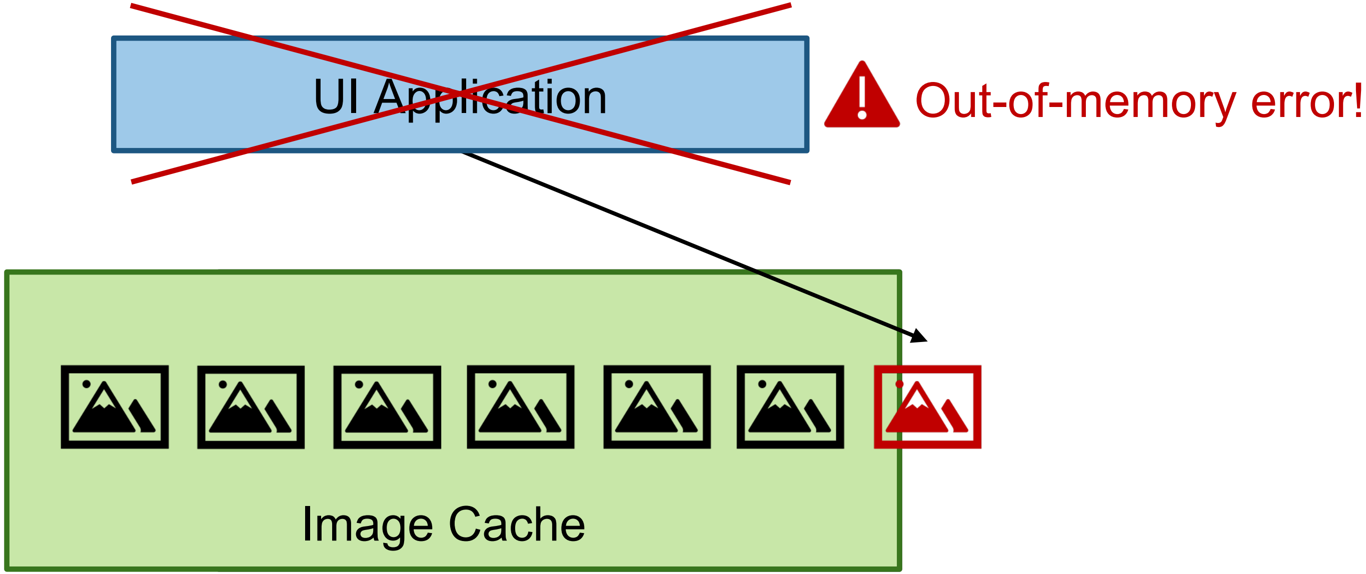
Cache

Available Memory

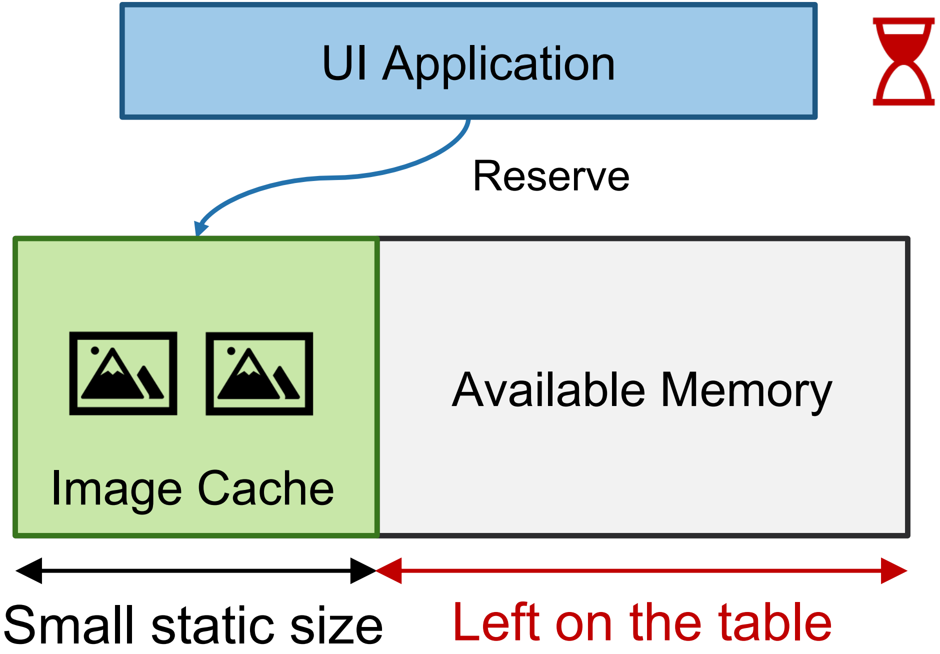
Option 1: Storing All Soft State



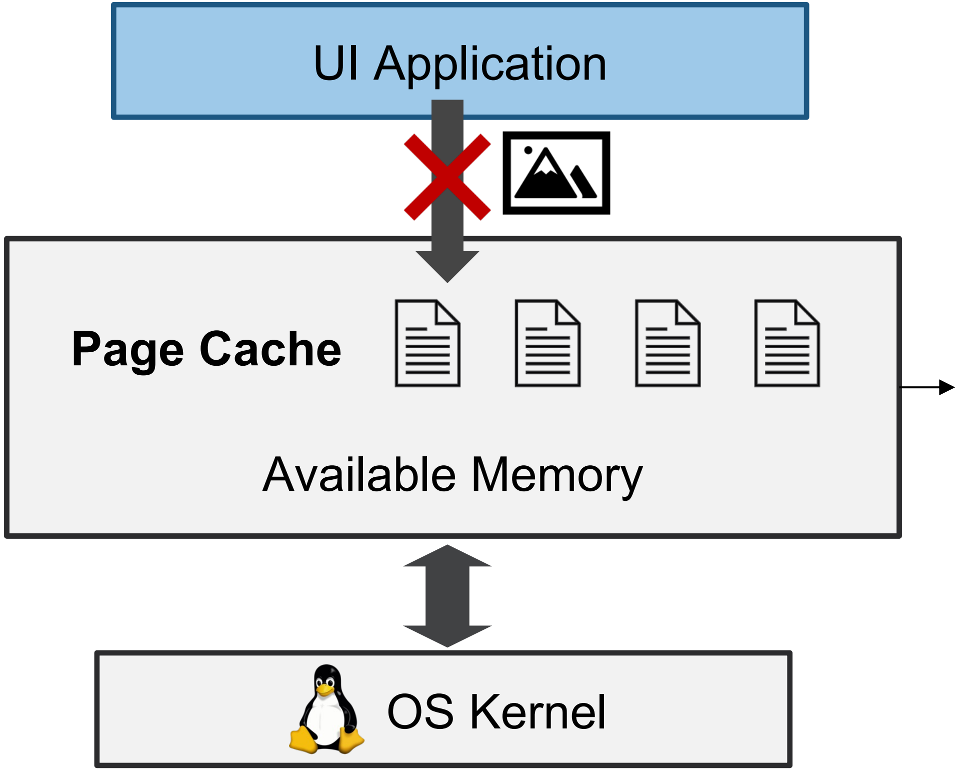
Option 1: Storing All Soft State



Option 2: Statically Limiting Cache Size



Option 3: Leveraging OS Page Cache



Limited to cache disk blocks

Design Goals

Option 1: storing all soft state

Option 2: static limit on cache size

Option 3: OS kernel page cache

Design Goals

~~Option 1: storing all soft state~~

~~Option 2: static limit on cache size~~

~~Option 3: OS kernel page cache~~

Responding to memory pressure

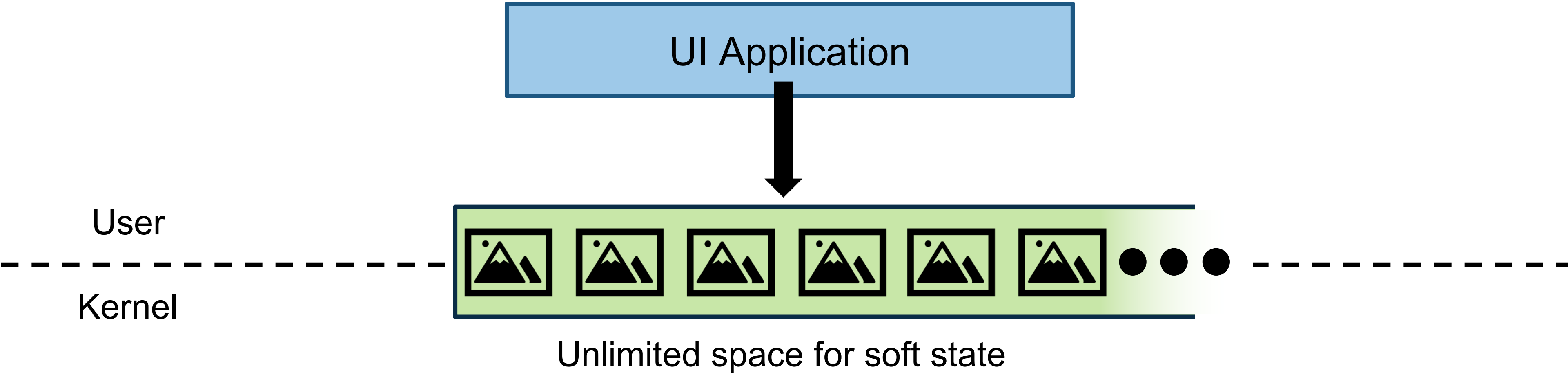
Taking full advantage of available memory

Democratizing what can be stored

Can we have a new virtual memory abstraction for soft state?

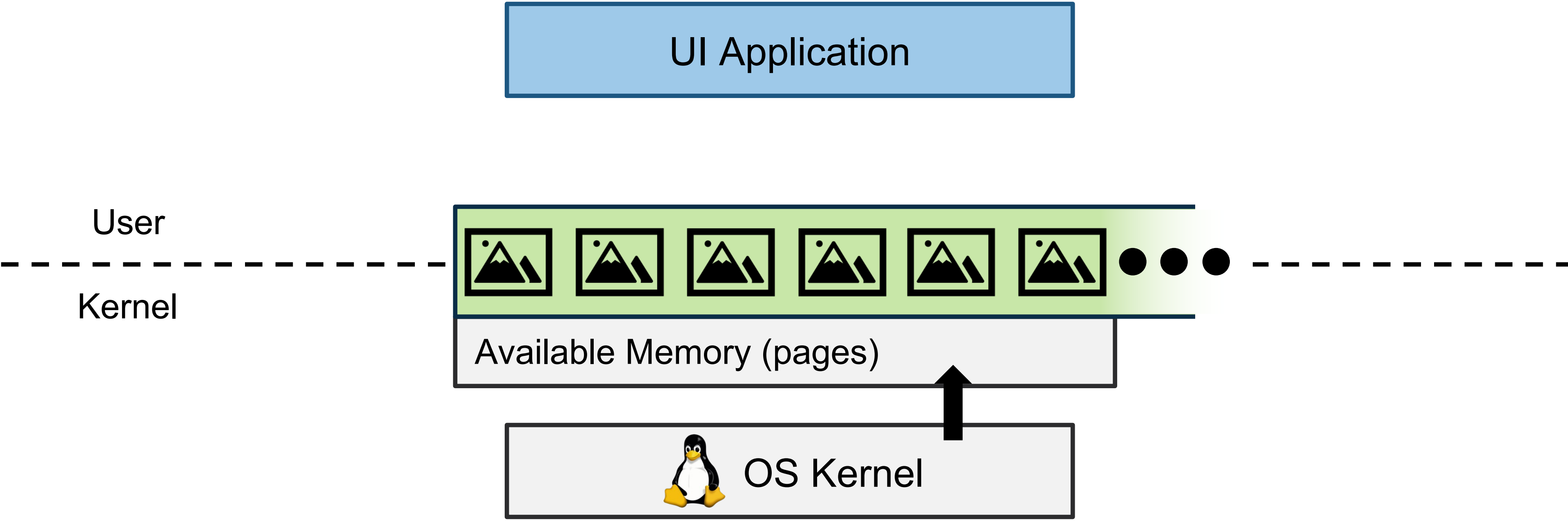
Midas: A Soft Memory Abstraction

① Offer the illusion of an **unlimited** cache space



Midas: A Soft Memory Abstraction

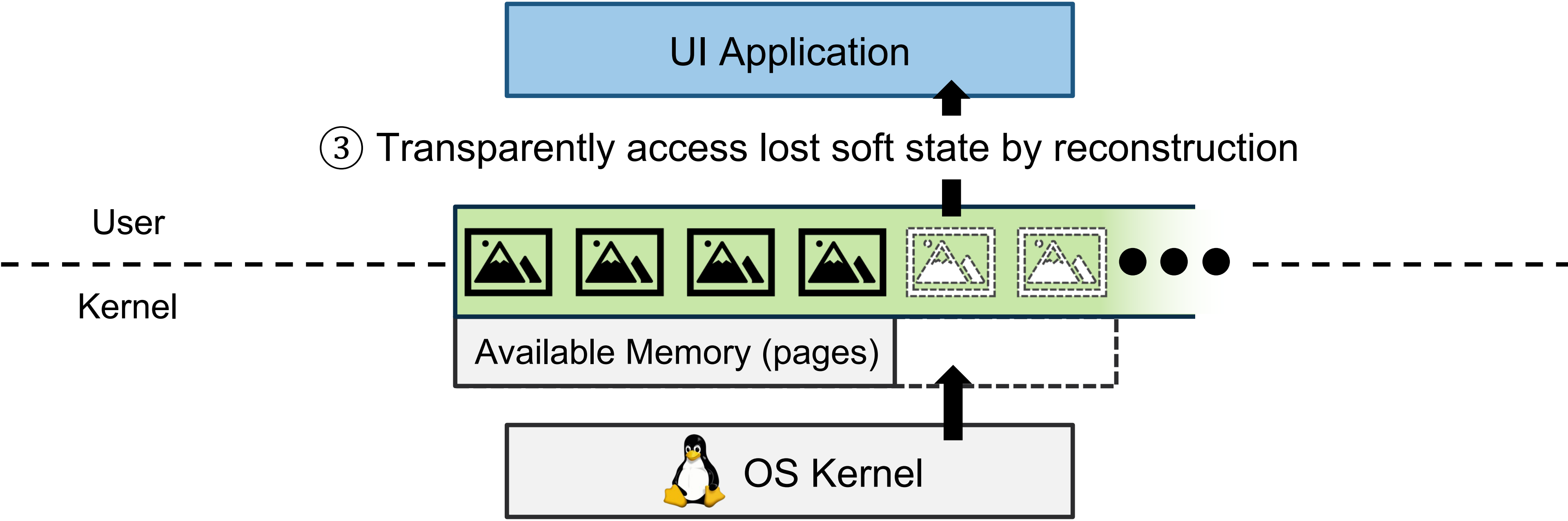
① Offer the illusion of an **unlimited** cache space



② Rapidly unmap memory pages to avoid running out of memory

Midas: A Soft Memory Abstraction

① Offer the illusion of an **unlimited** cache space



③ Transparently access lost soft state by reconstruction

② Rapidly unmap memory pages to avoid running out of memory

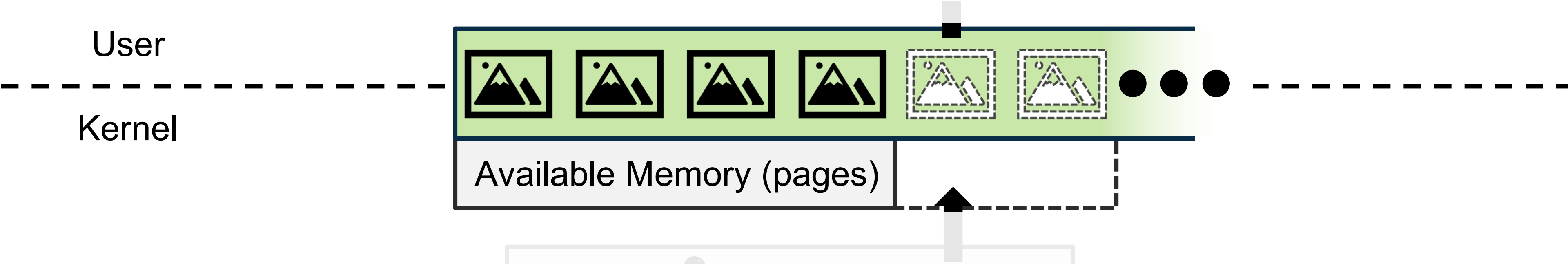
Midas: A Soft Memory Abstraction

① Offer the illusion of an **unlimited** cache space

UI Application

How to access soft memory?

③ Transparently access lost soft state by reconstruction



OS Kernel

How to reclaim soft memory?

② Rapidly unmap memory pages to avoid running out of memory

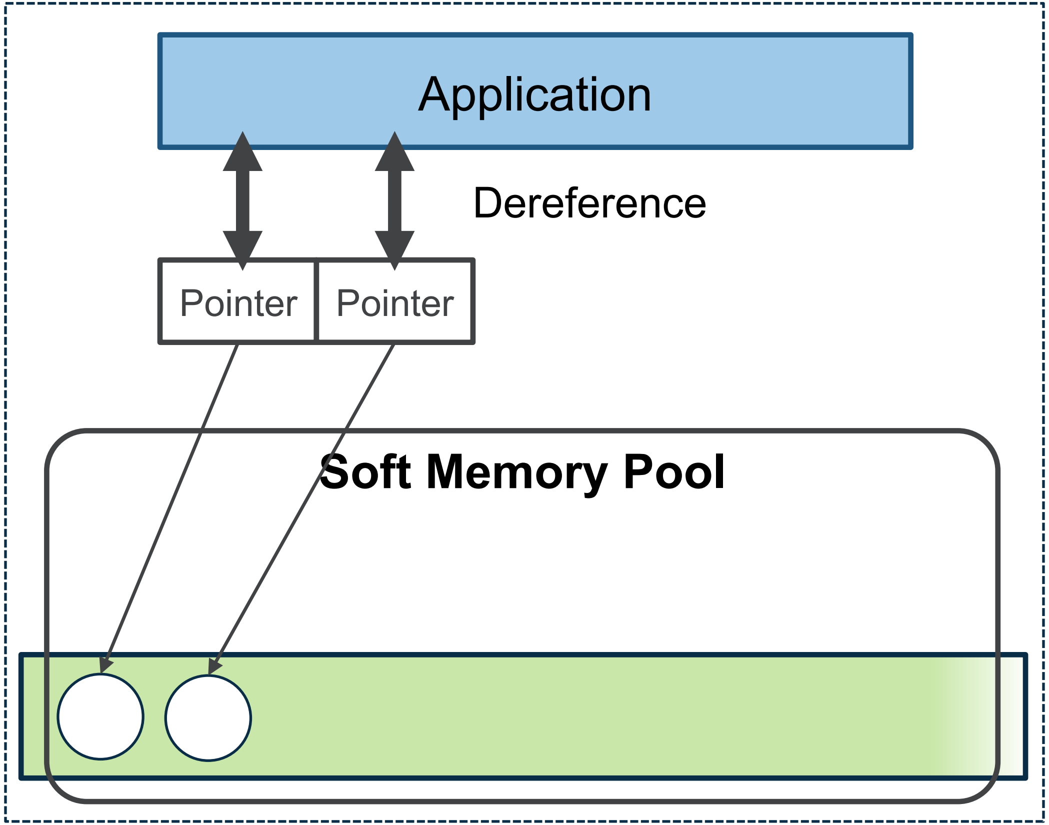
How to Access Soft Memory?

Soft memory pointers

➢ Similar to smart pointers

Soft memory pool

- Allocator



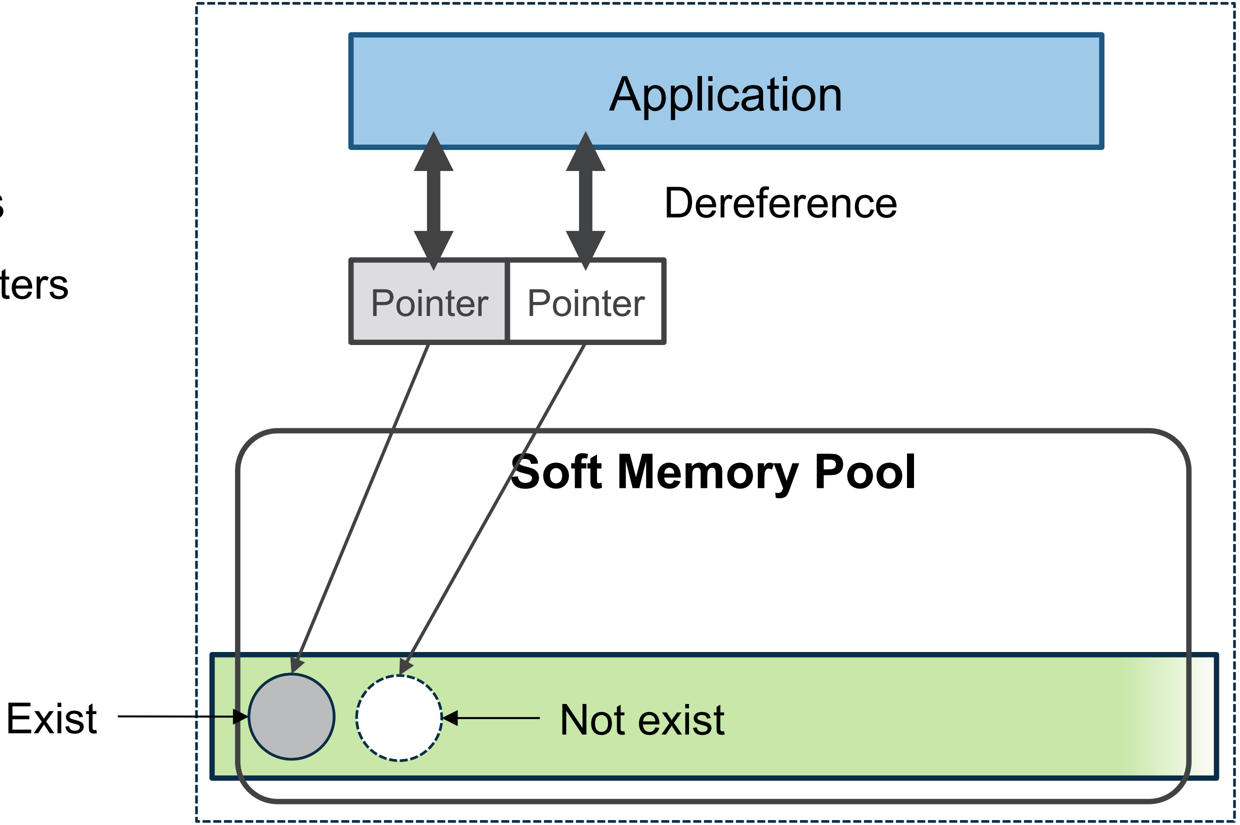
How to Access Soft Memory?

Soft memory pointers

- Similar to smart pointers

Soft memory pool

- Allocator



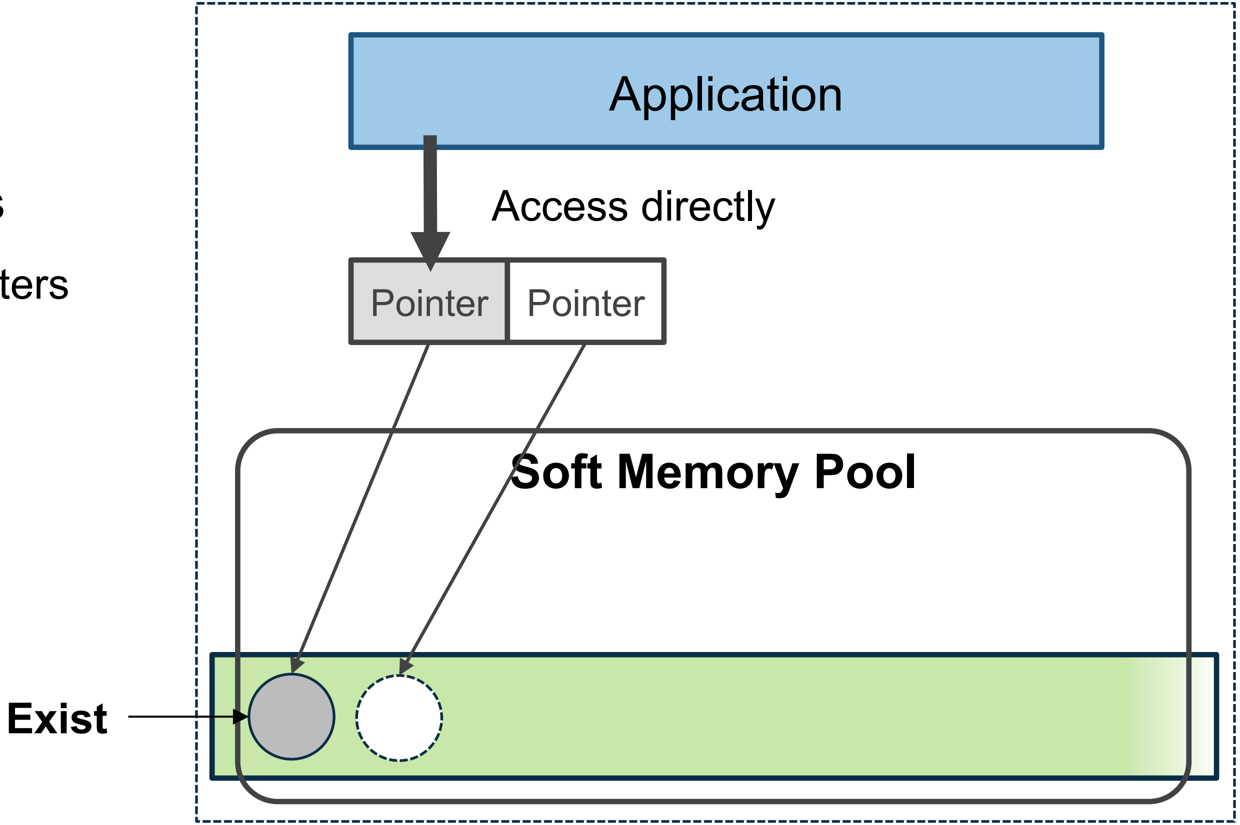
How to Access Soft Memory?

Soft memory pointers

- Similar to smart pointers

Soft memory pool

- Allocator



How to Access Soft Memory?

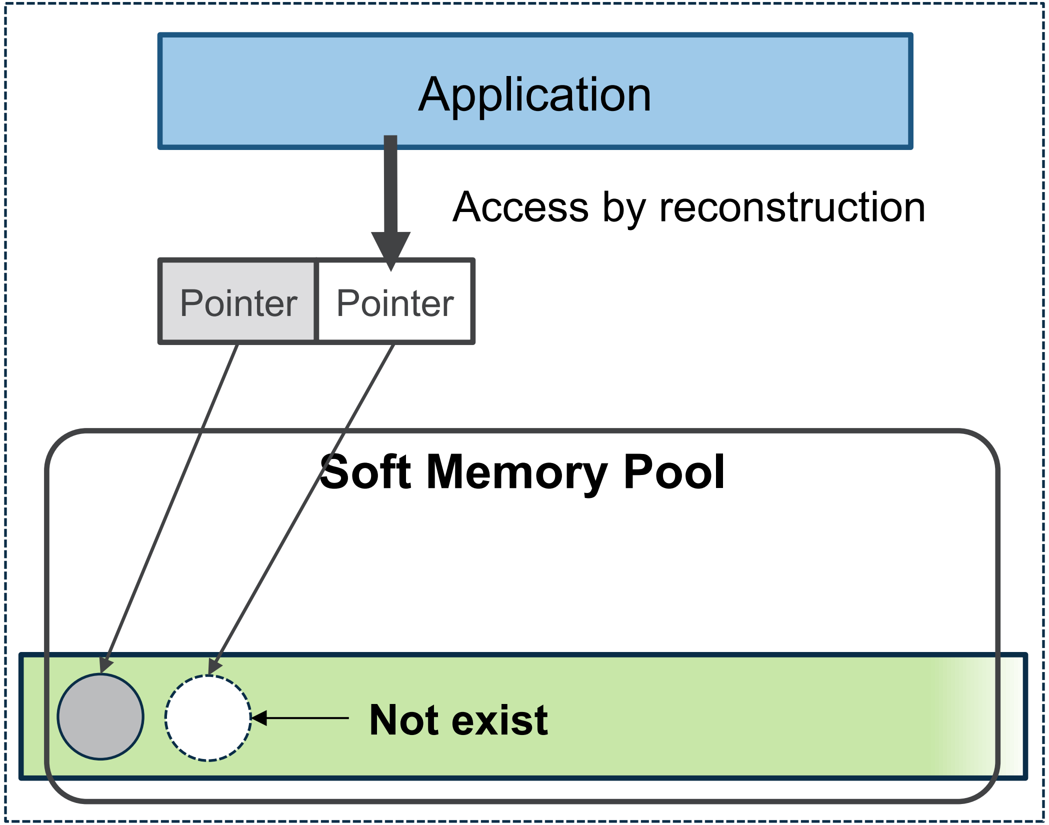
Soft memory pointers

- Similar to smart pointers

➢ Transparent reconstruction

Soft memory pool

- Allocator



How to Transparently Reconstruct Soft State?

Soft memory pointers

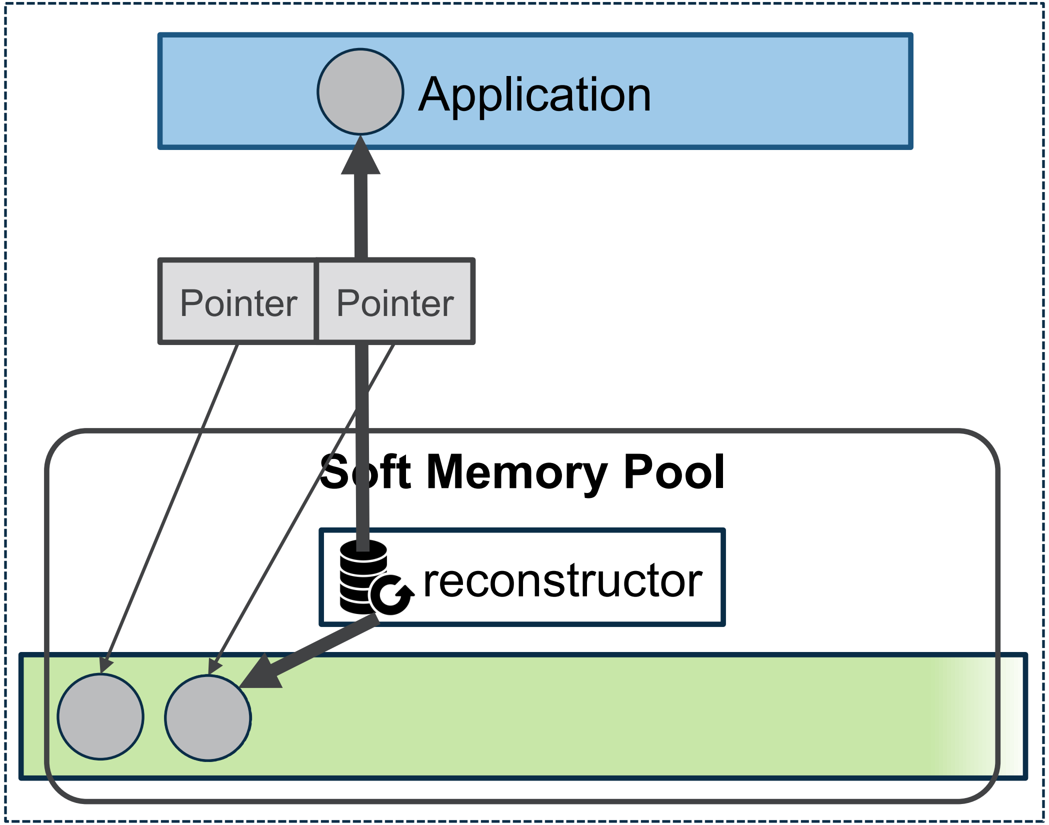
- Similar to smart pointers

➢ Transparent reconstruction

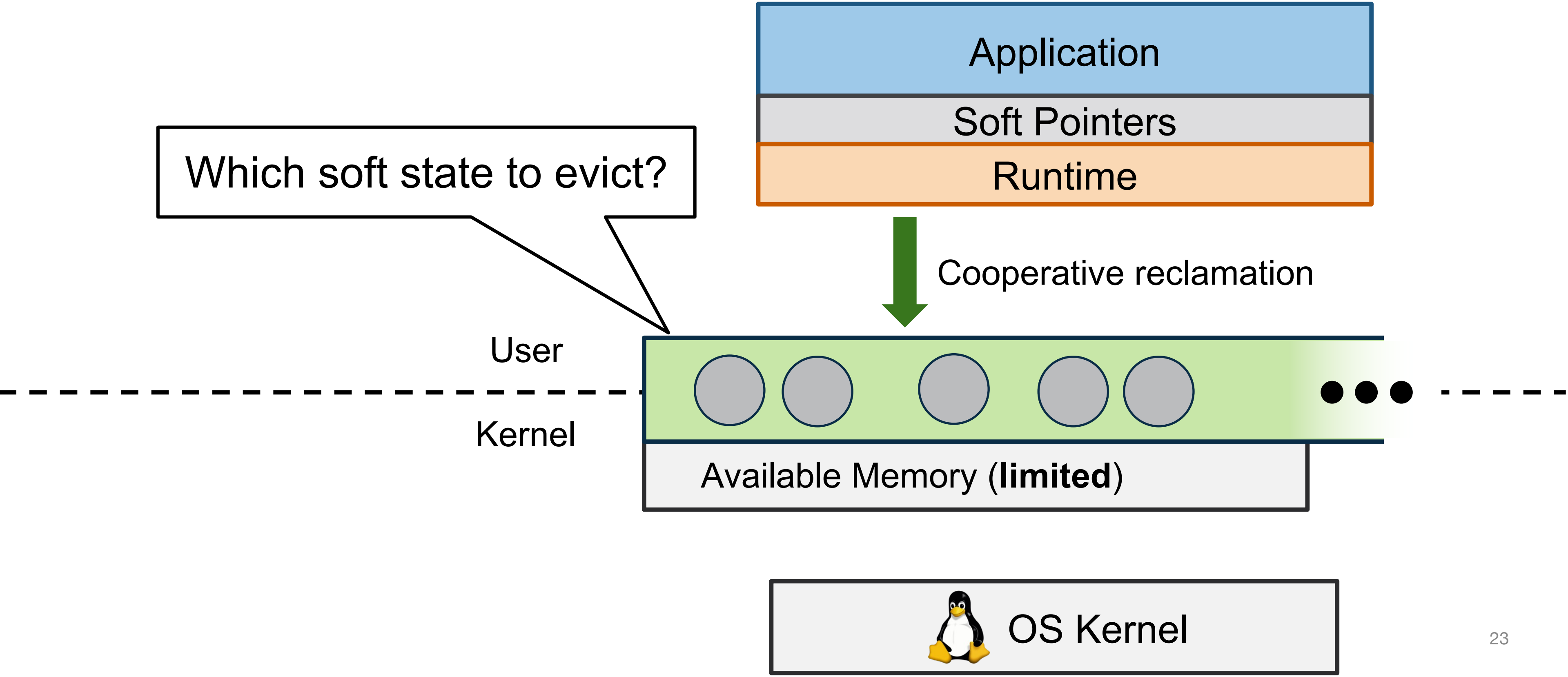
Soft memory pool

- Allocator

➢ Initialized with reconstruction logic



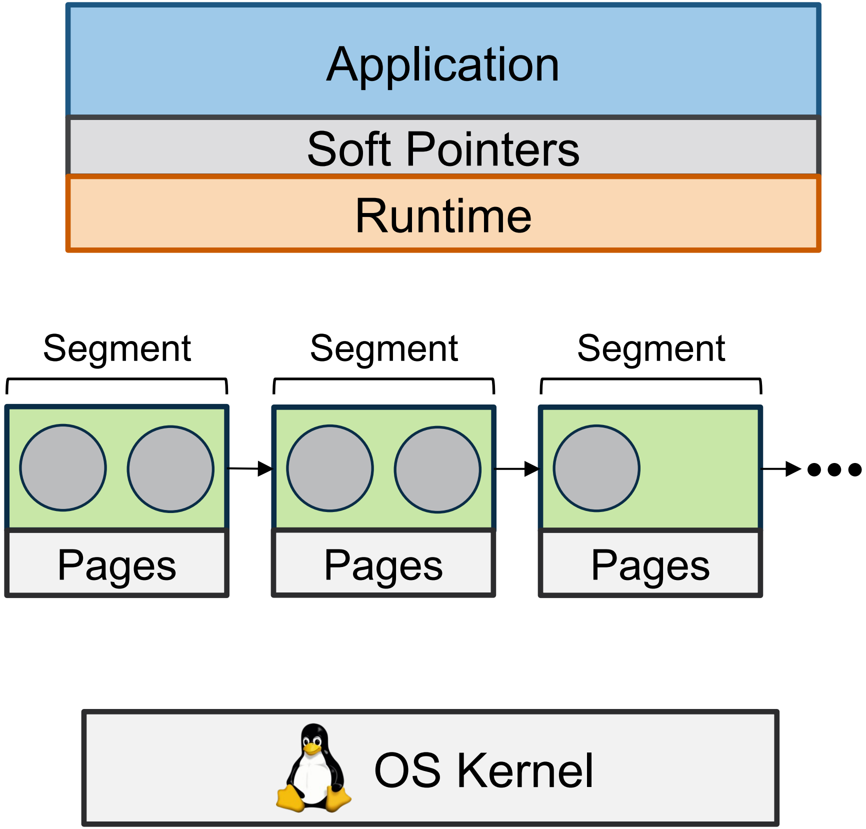
How to Reclaim Soft Memory?



Runtime Memory Management

Log-structured allocator

- ➢ Organize soft memory as segments

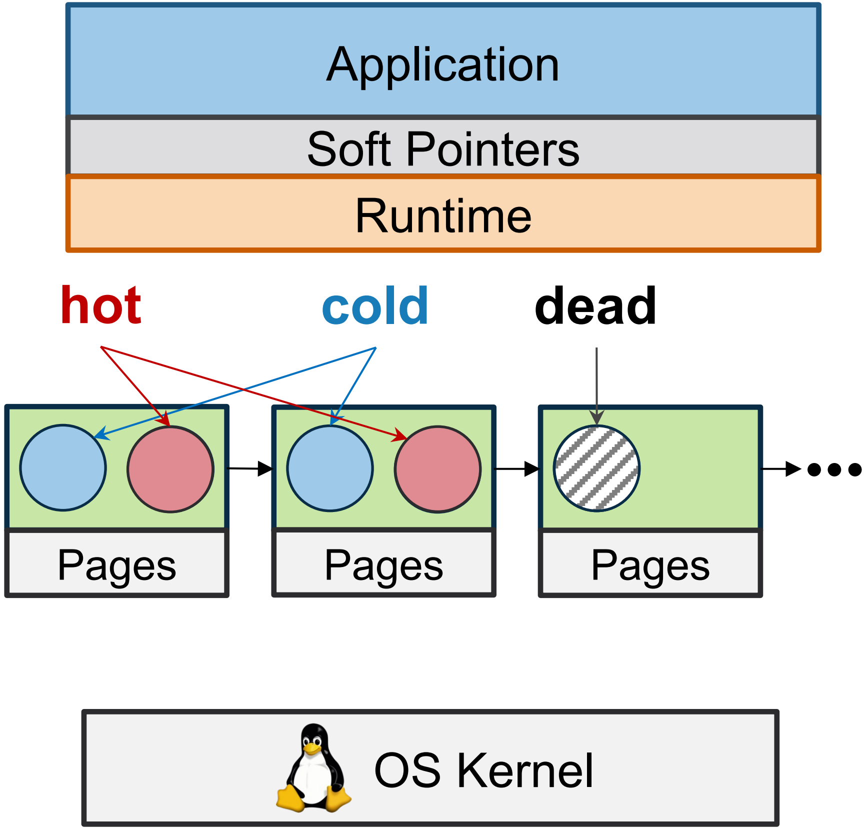


Runtime Memory Management

Log-structured allocator

- Organize soft memory as segments

➢ Track access frequency (hotness)



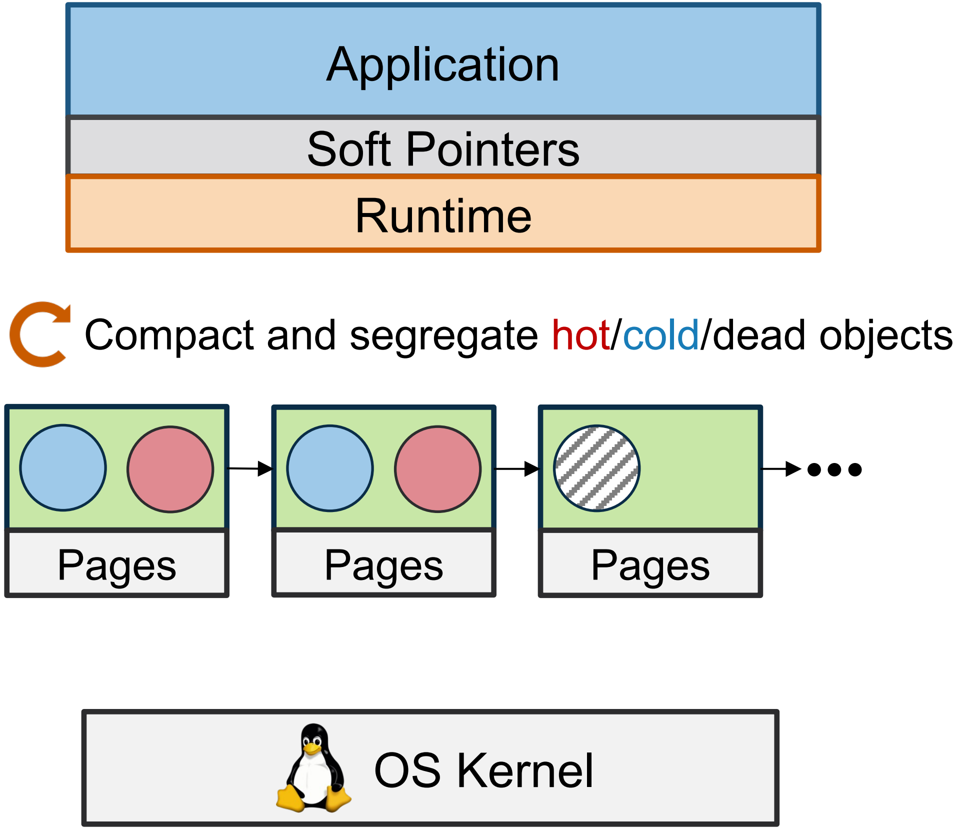
Runtime Memory Management

Log-structured allocator

- Organize soft memory as segments
- Track access frequency (hotness)

Concurrent evacuator

➢ Continuously compact objects



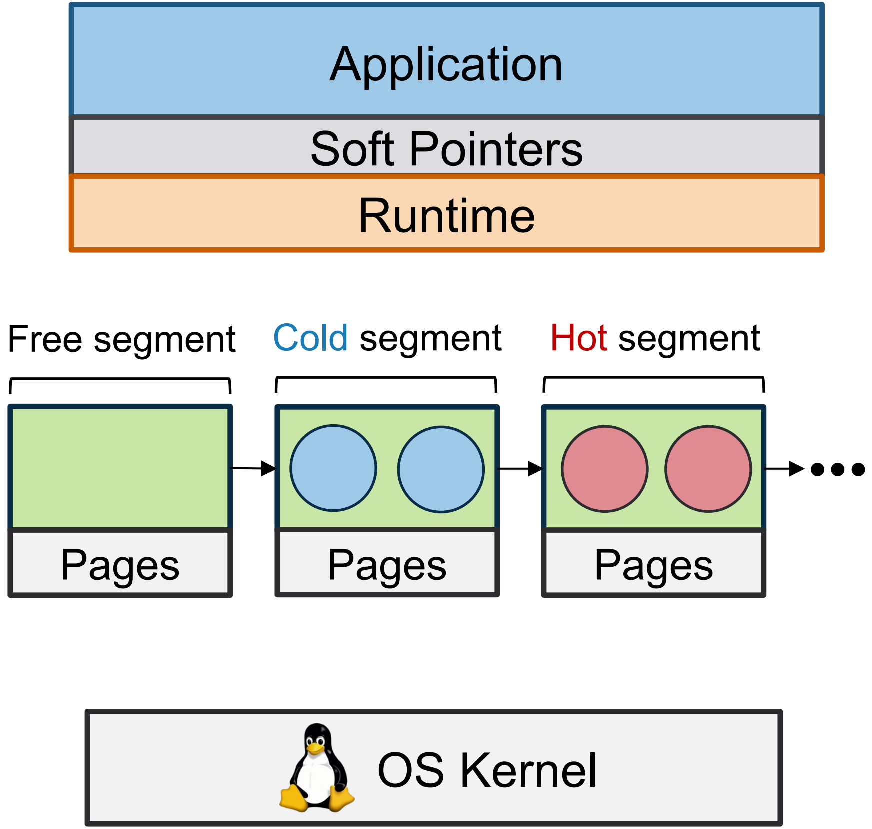
Runtime Memory Management

Log-structured allocator

- Organize soft memory as segments
- Track access frequency (hotness)

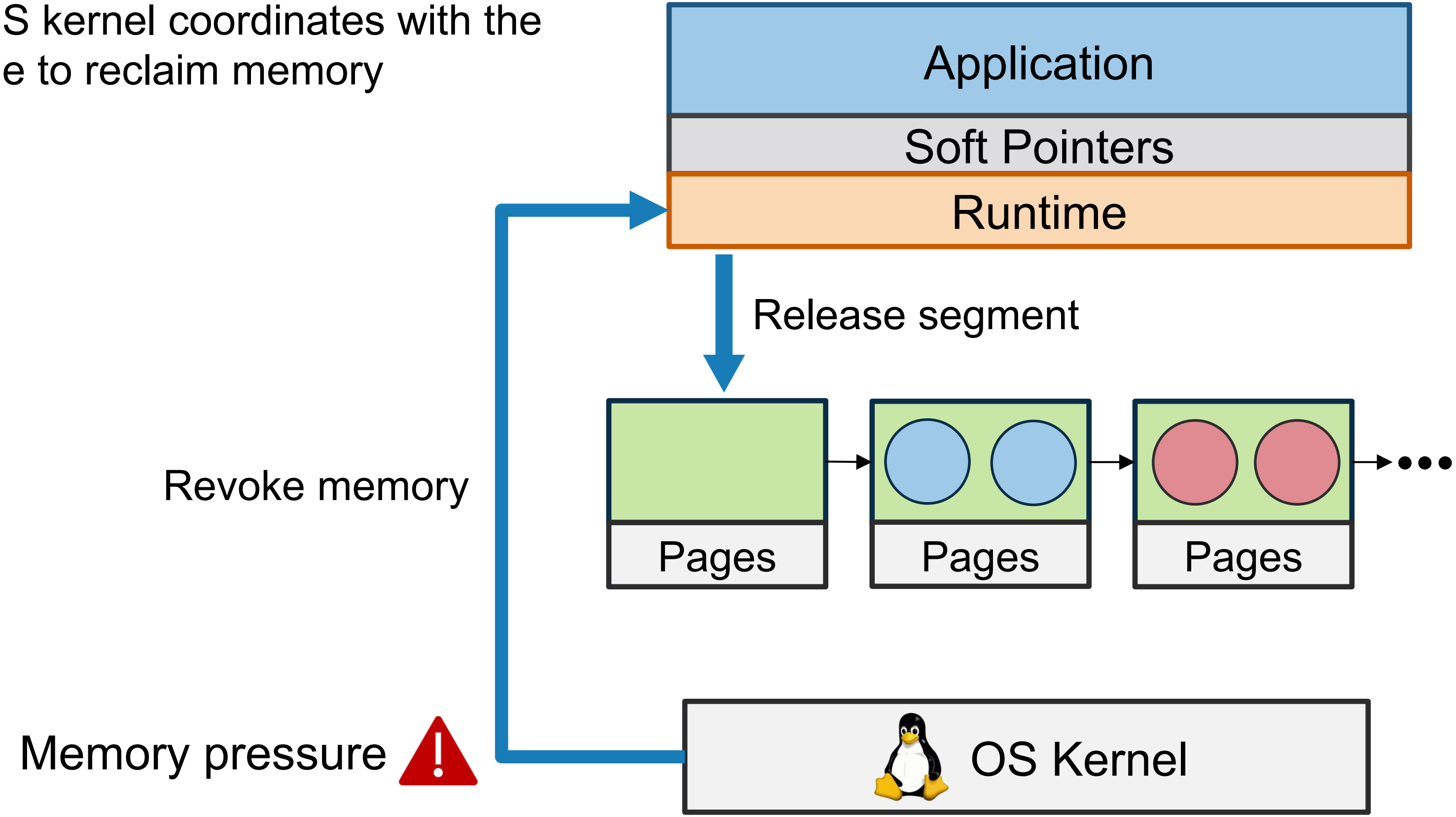
Concurrent evacuator

- ➢ Continuously compact objects



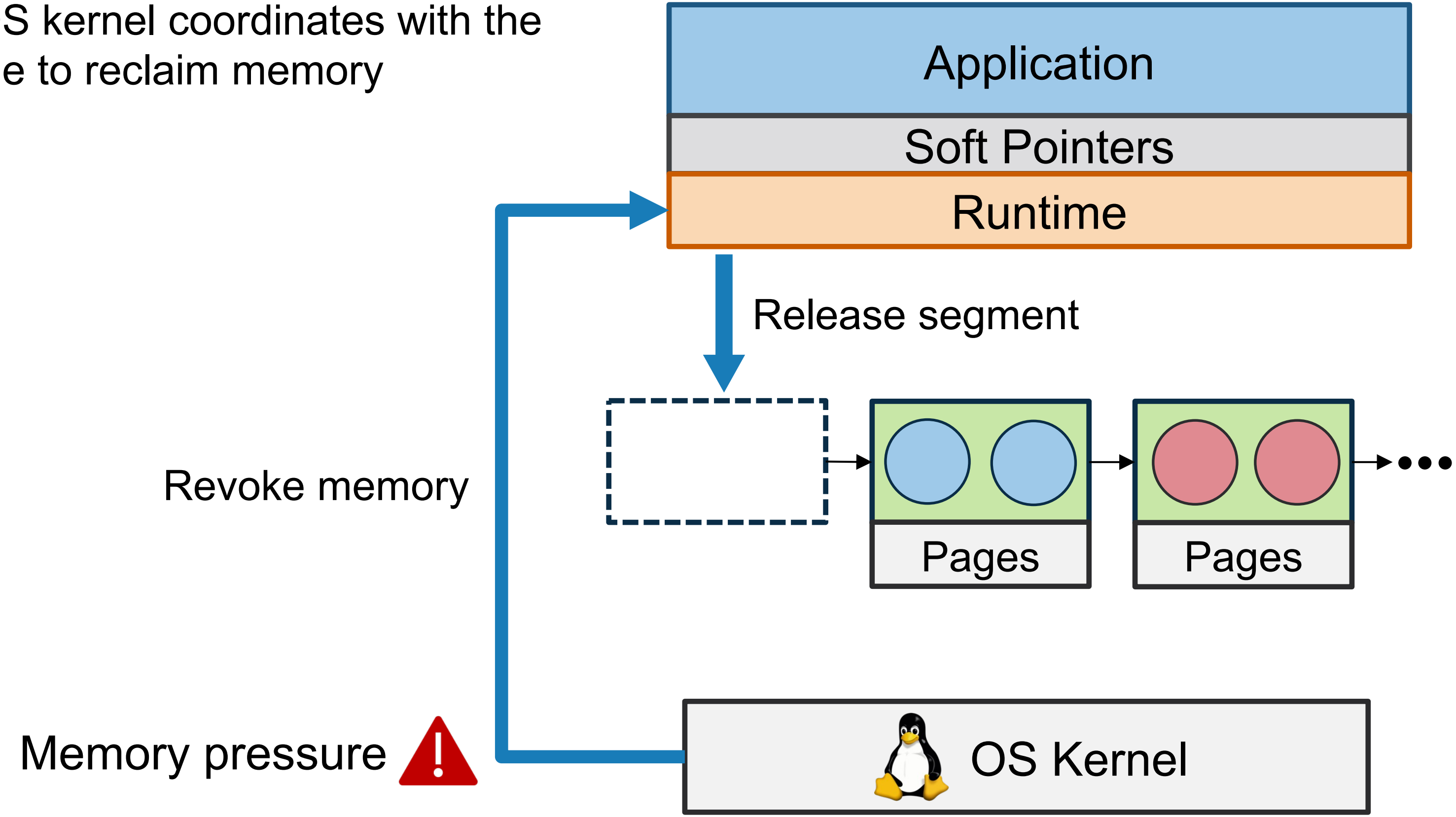
Runtime Cooperative Reclamation

The OS kernel coordinates with the runtime to reclaim memory



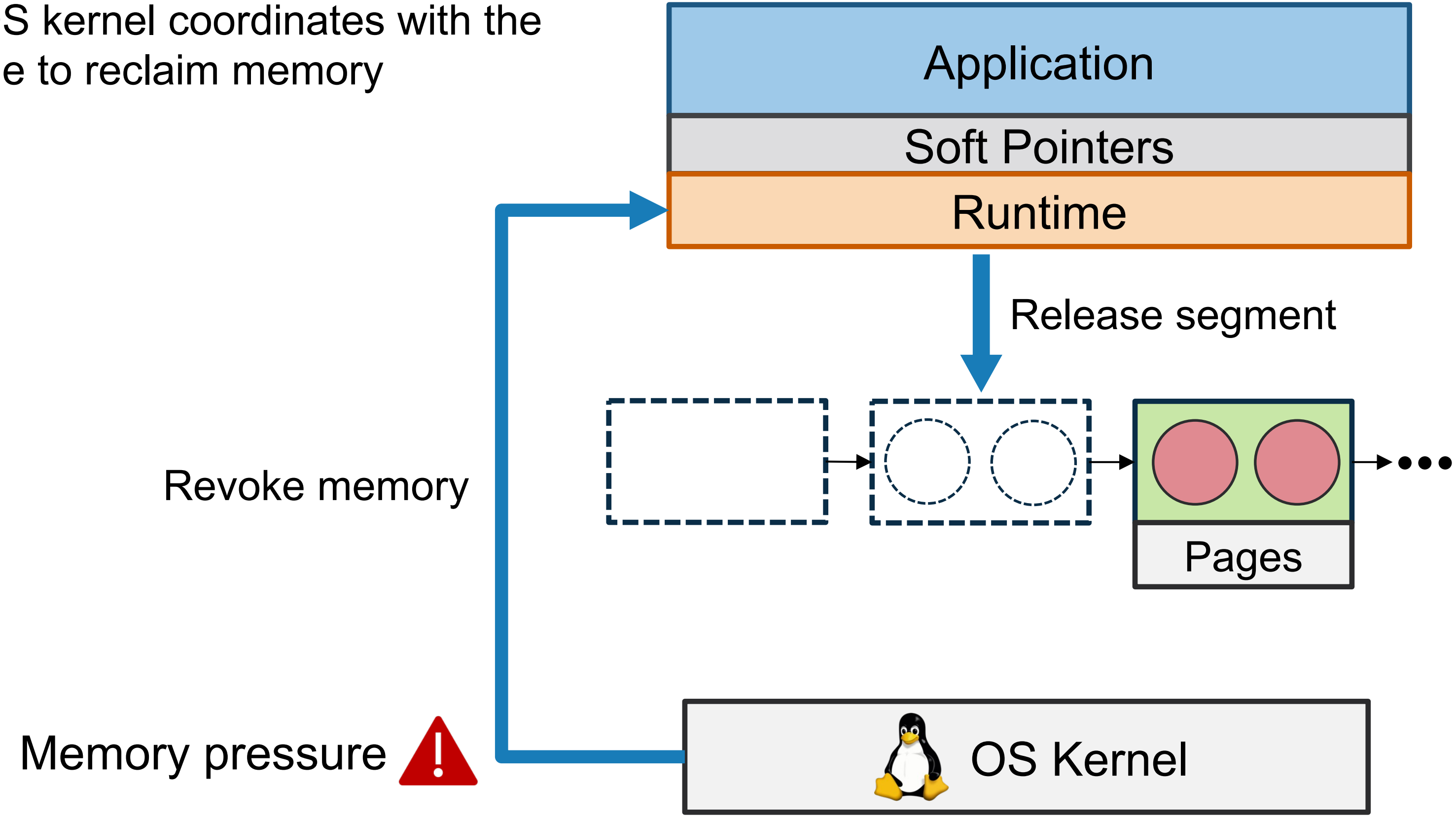
Runtime Cooperative Reclamation

The OS kernel coordinates with the runtime to reclaim memory



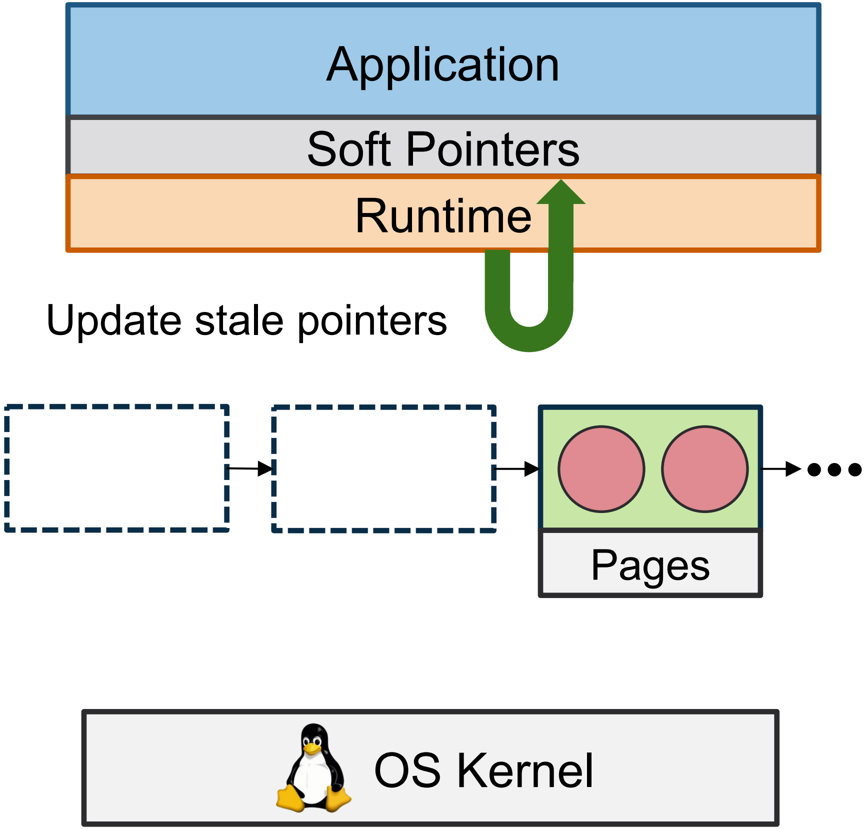
Runtime Cooperative Reclamation

The OS kernel coordinates with the runtime to reclaim memory



Runtime Cooperative Reclamation

The OS kernel coordinates with the runtime to reclaim memory

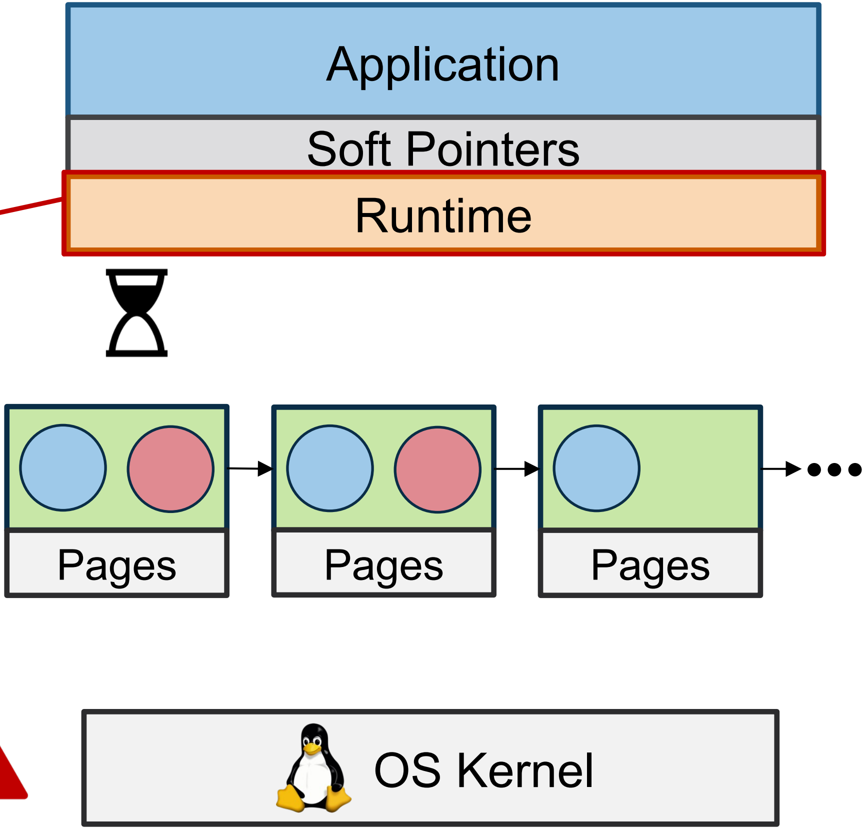


Memory pressure ✓

Runtime Cooperative Reclamation

The OS kernel coordinates with the runtime to reclaim memory

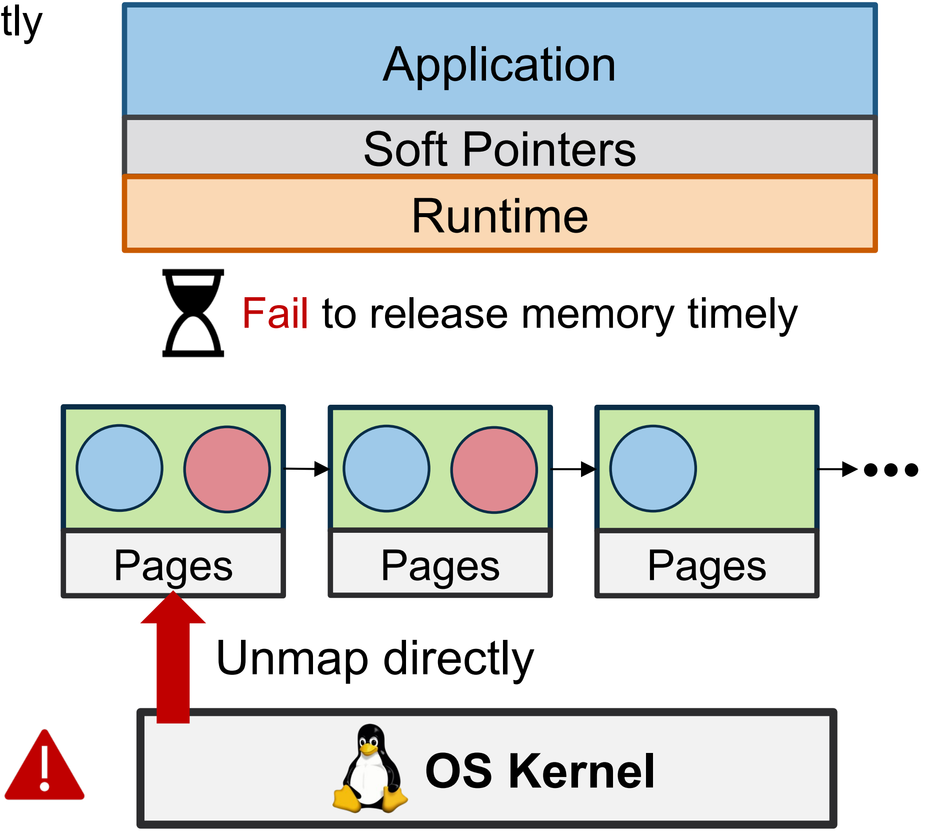
What if the runtime fails to release memory timely?



Severe memory pressure ⚠

Kernel Enforced Reclamation

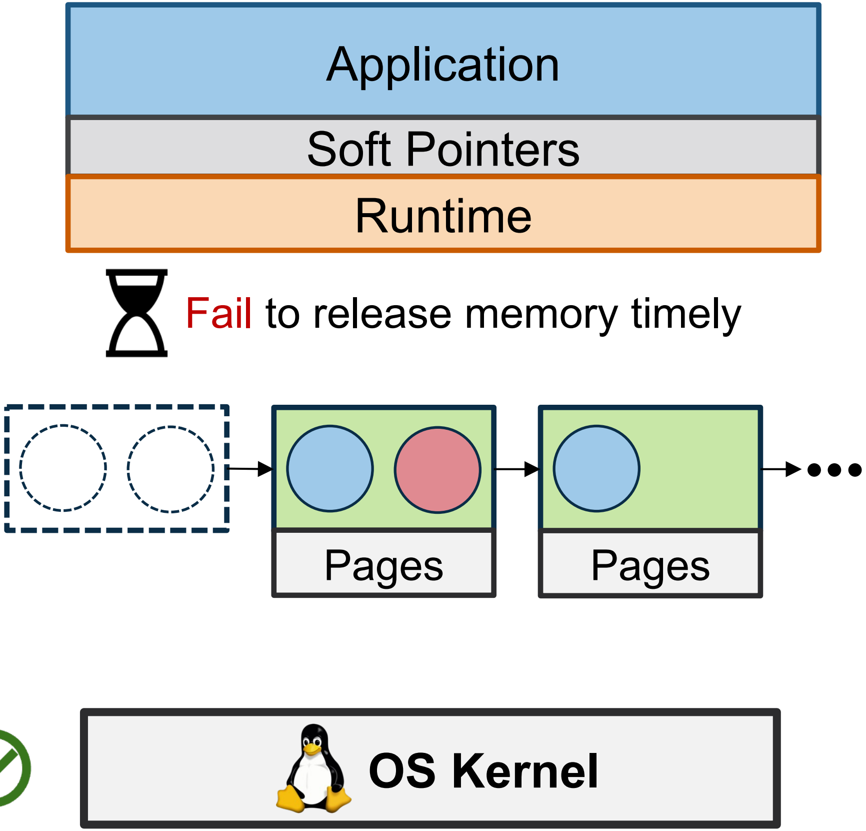
The OS kernel unmaps pages directly



Severe memory pressure ⚠

Kernel Enforced Reclamation

The OS kernel unmaps pages directly



Application

Soft Pointers

Runtime

Fail to release memory timely

Pages

Pages

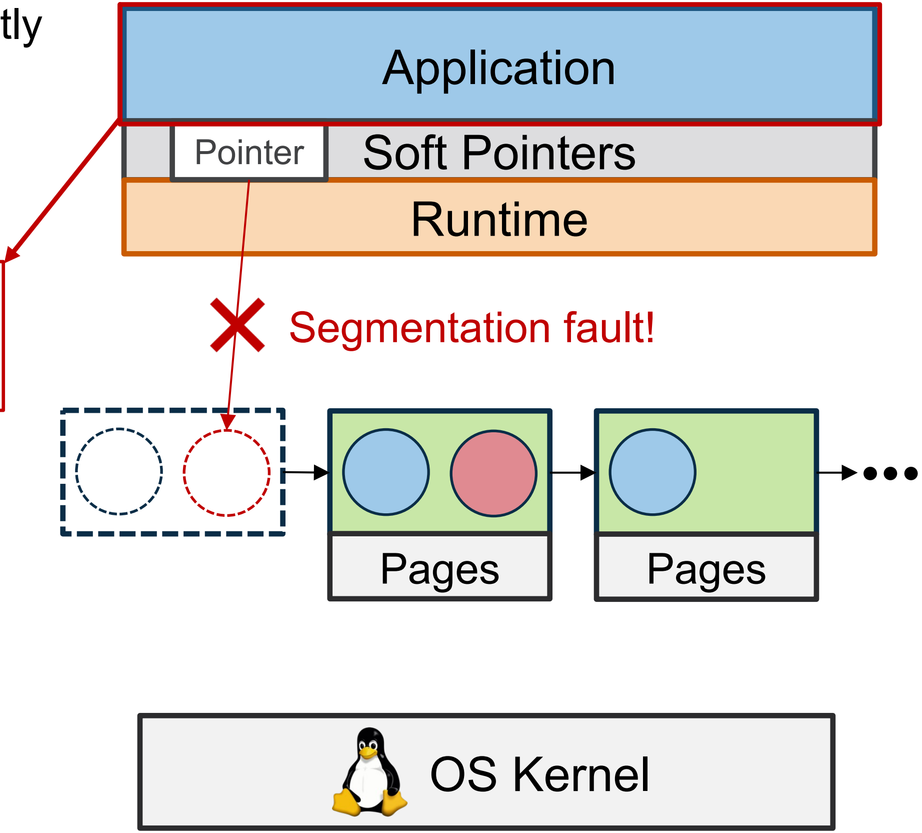
OS Kernel

Severe memory pressure ✓

Kernel Enforced Reclamation

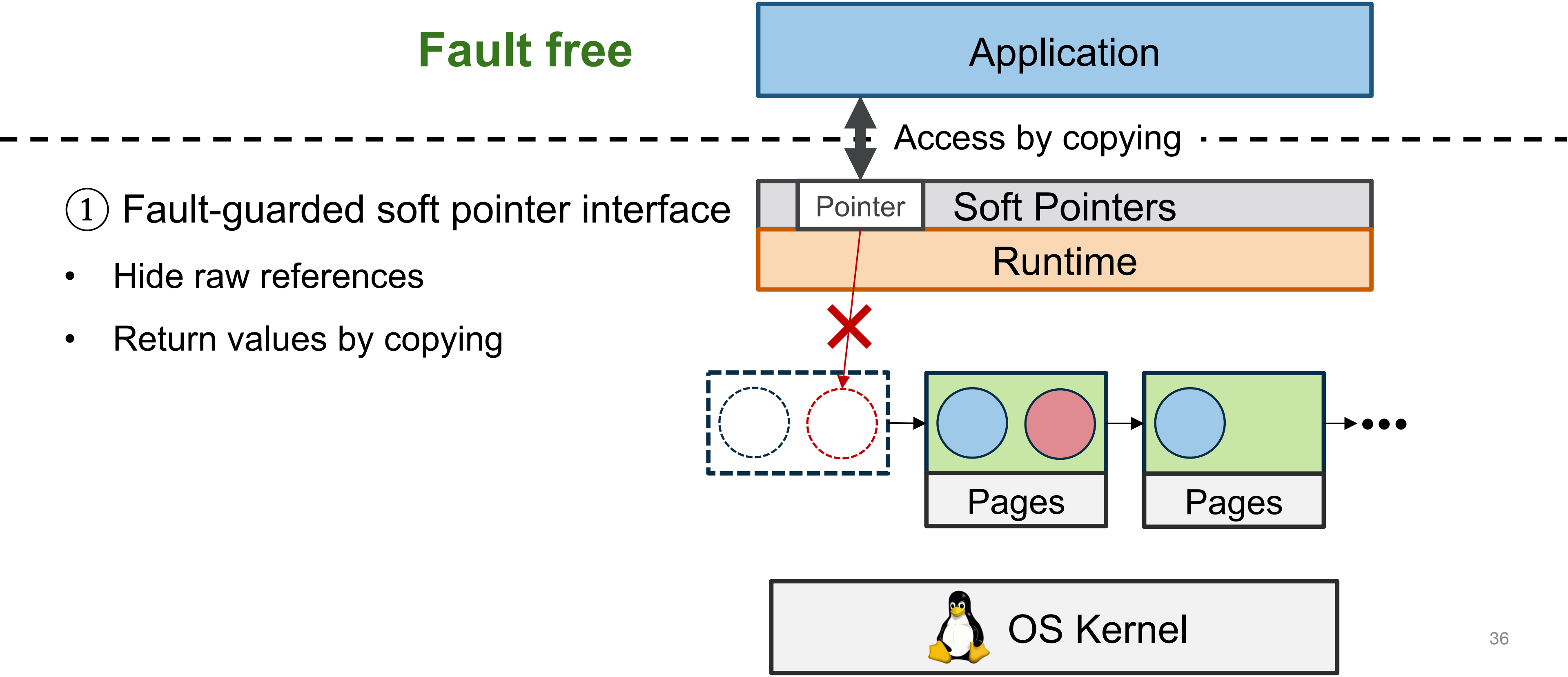
The OS kernel unmaps pages directly

How to protect the application from segmentation faults?



How to Protect the Application From Segfaults?

Fault free



① Fault-guarded soft pointer interface

- Hide raw references
- Return values by copying

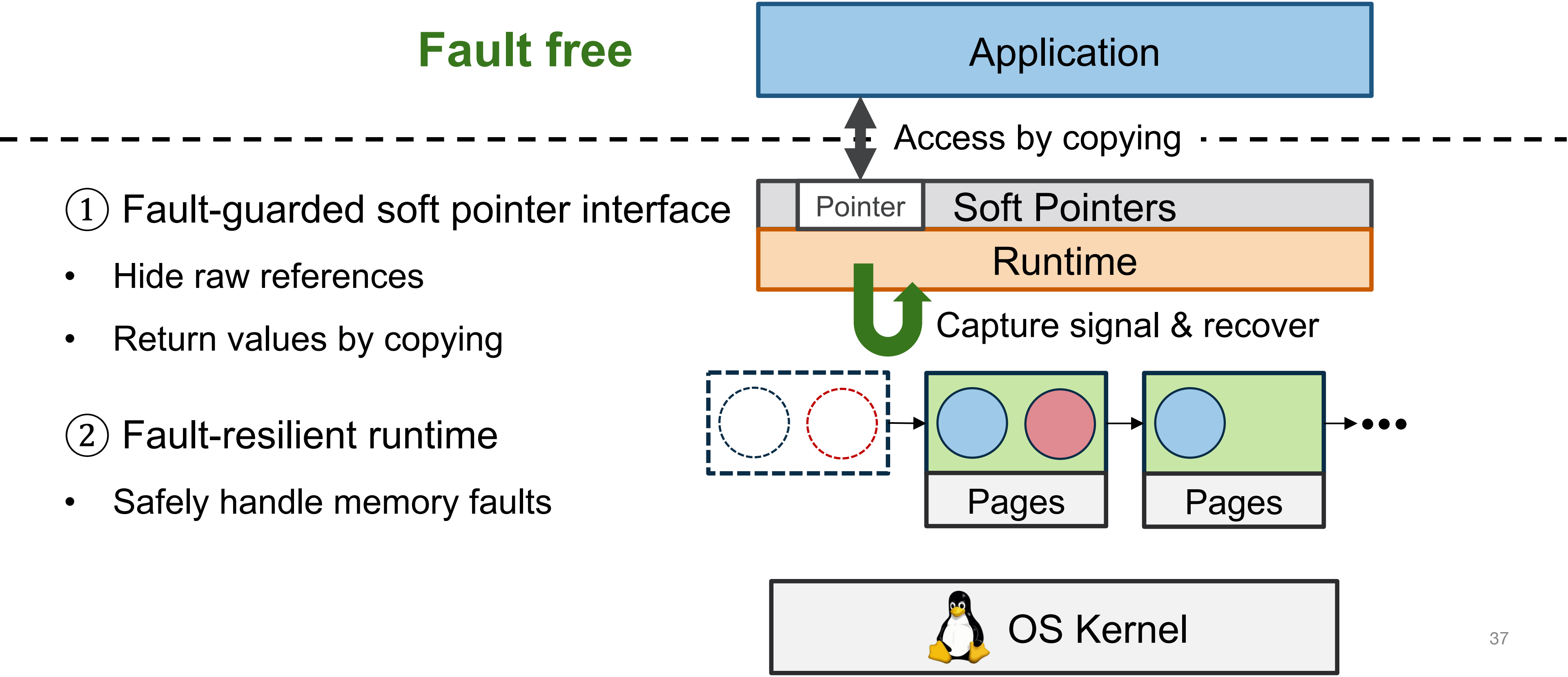
How to Protect the Application From Segfaults?

① Fault-guarded soft pointer interface

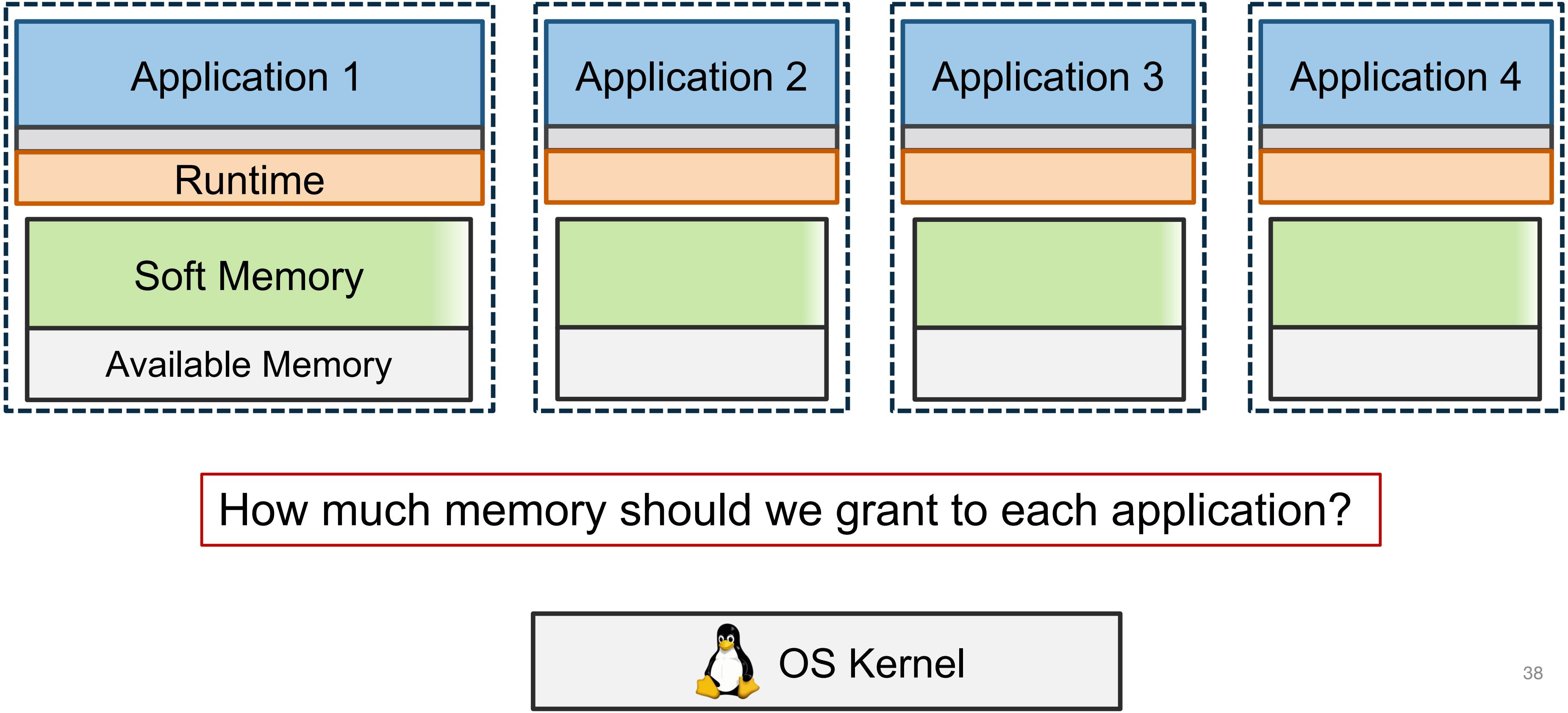
- Hide raw references
- Return values by copying

② Fault-resilient runtime

- Safely handle memory faults

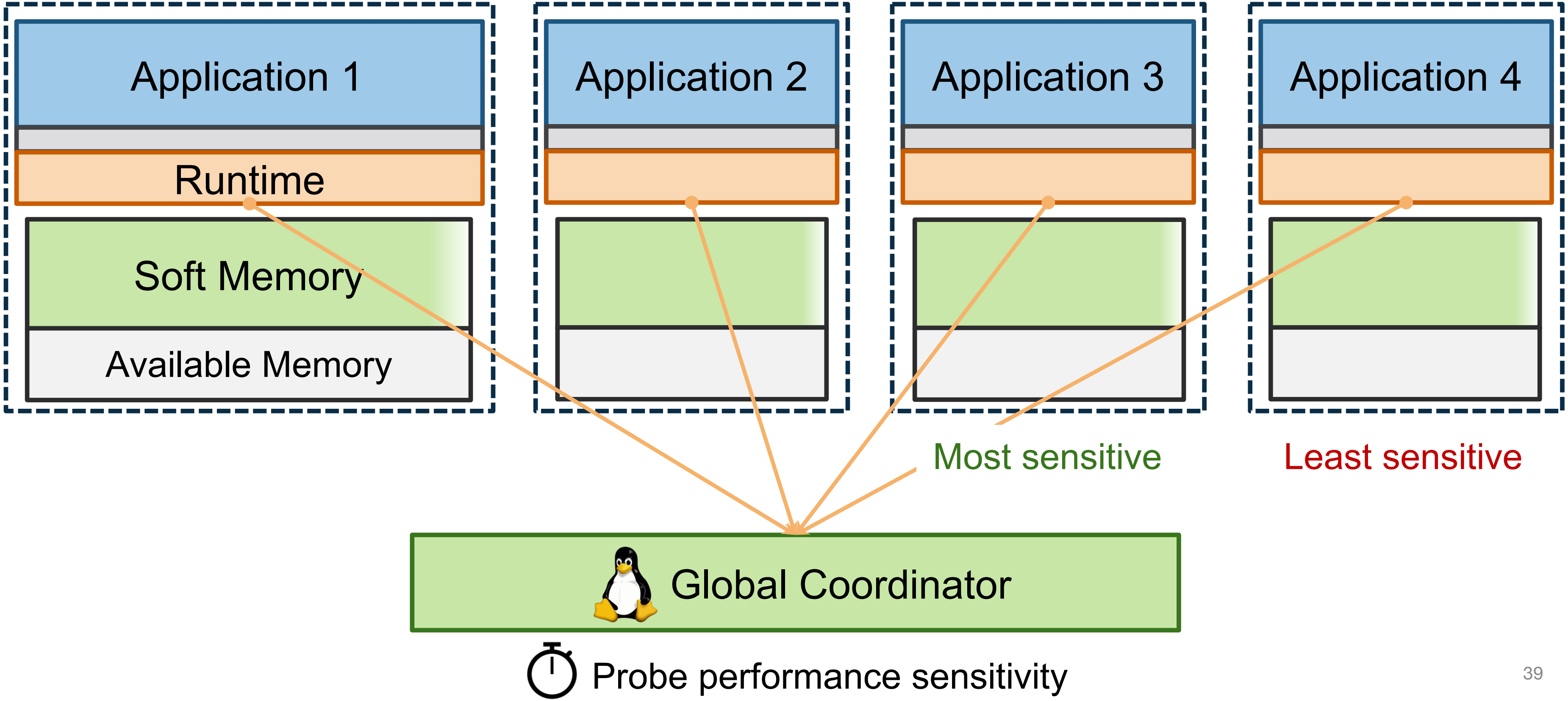


Midas in Practice

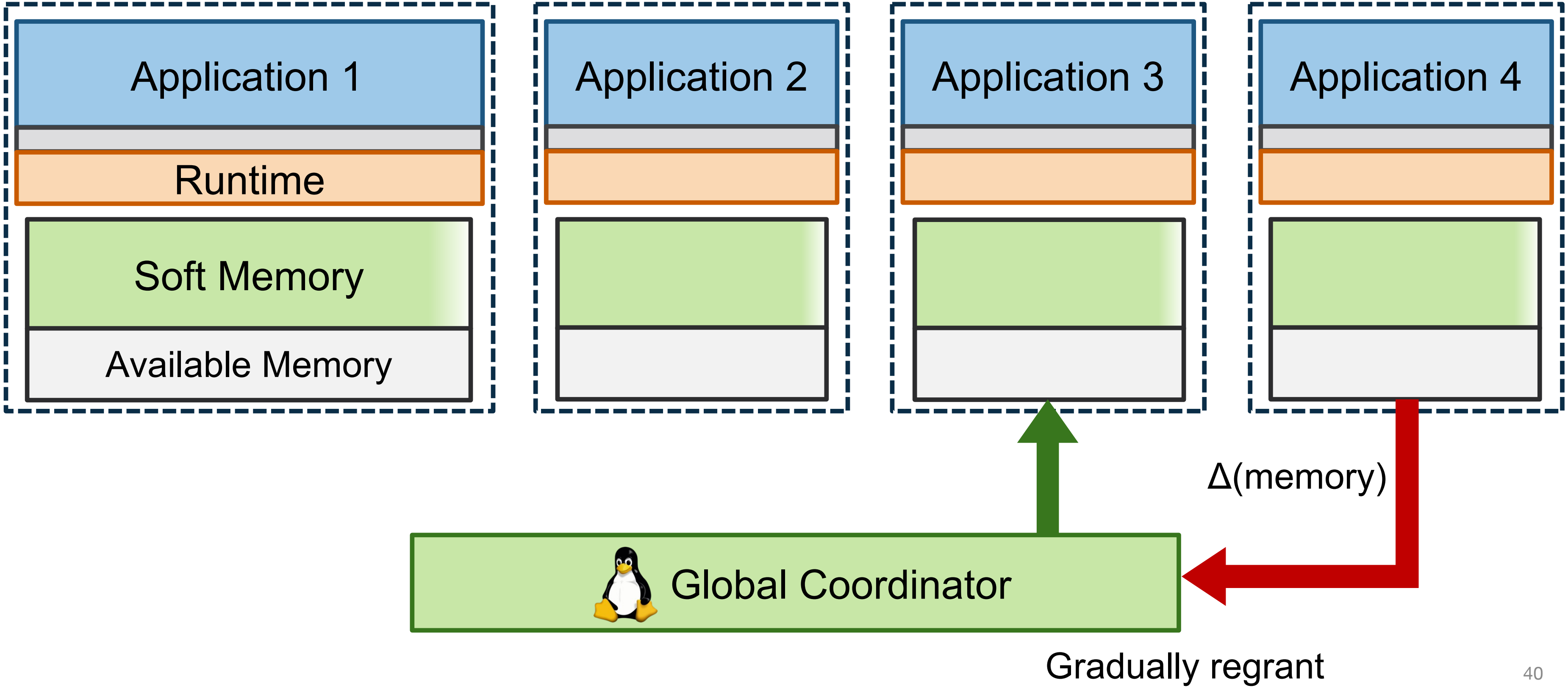


How much memory should we grant to each application?

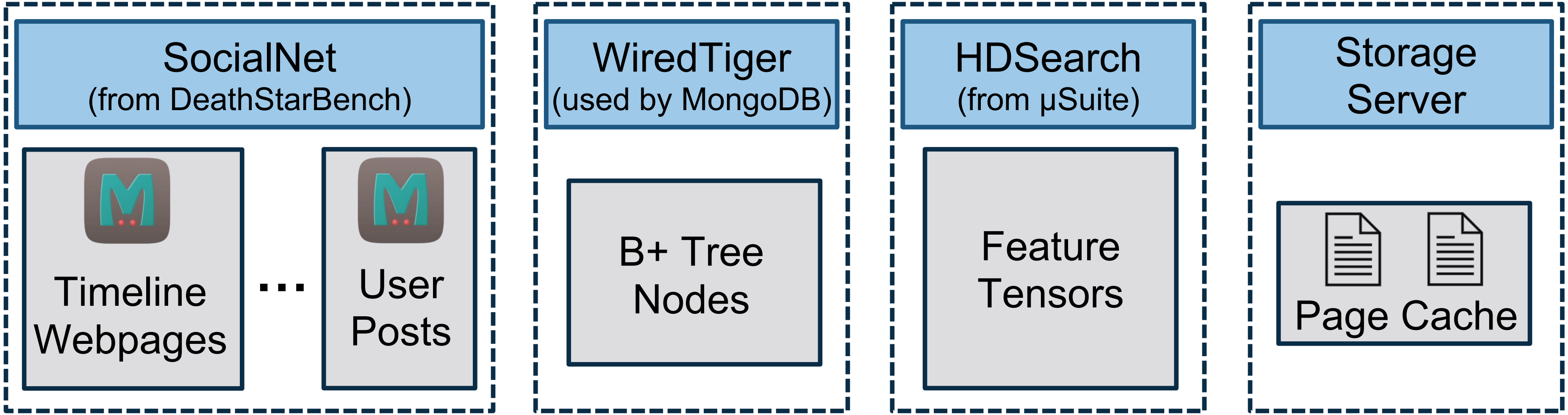
How to Coordinate Soft Memory Between Apps?



How to Coordinate Soft Memory Between Apps?



Midas in Practice



SocialNet (from DeathStarBench)

Timeline Webpages … User Posts

WiredTiger (used by MongoDB)

B+ Tree Nodes

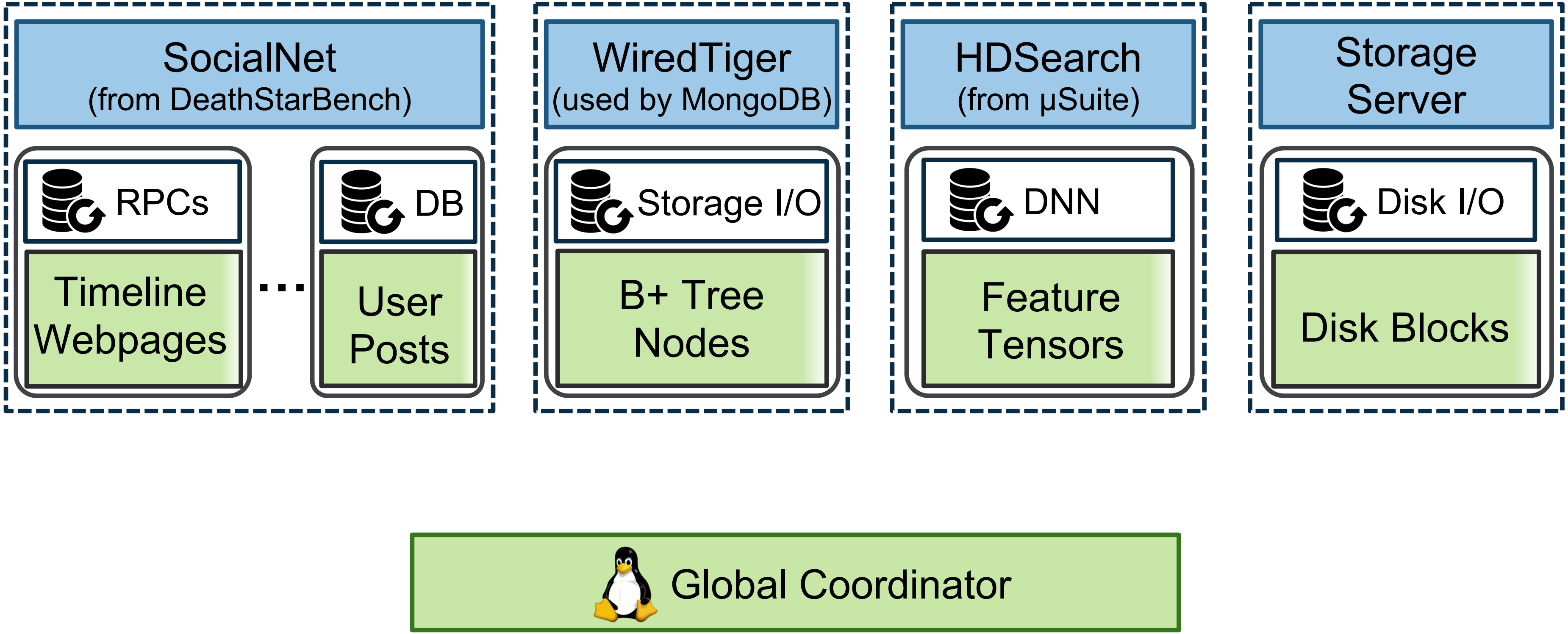
HDSearch (from µSuite)

Feature Tensors

Storage Server

Page Cache

Midas in Practice



Evaluation

SocialNet
(from DeathStarBench)

WiredTiger
(used by MongoDB)

HDSearch
(from µSuite)

Storage
Server

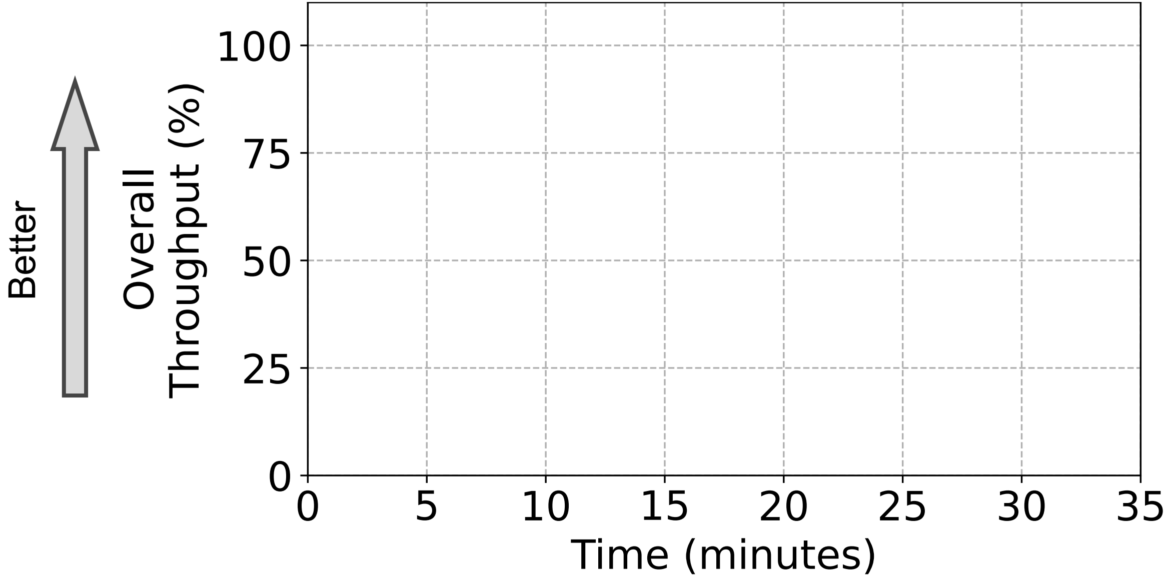
1. Can Midas harvest and coordinate soft memory among applications?

2. Can Midas quickly react to memory pressure?

Global Coordinator

Colocating Four Applications

- 20 GiB idle memory



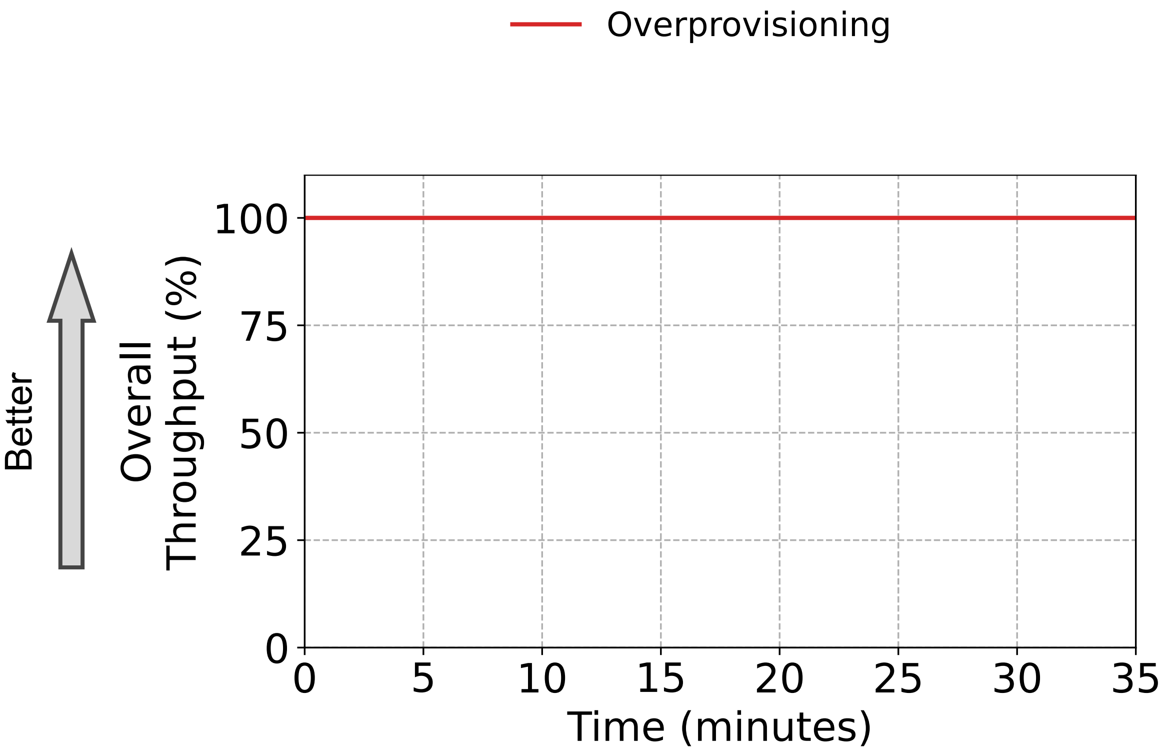
Colocating Four Applications

- 20 GiB idle memory

Baselines:

1. **Overprovisioning**

   (67.5 GiB soft memory usage)

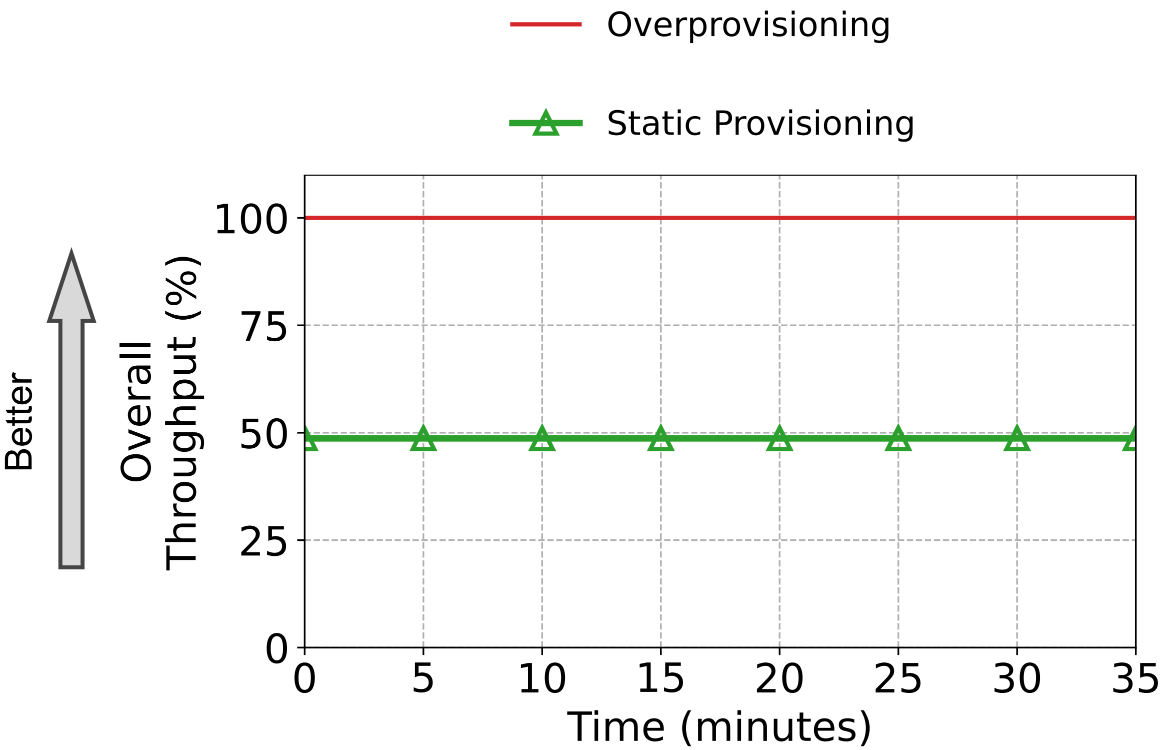


Colocating Four Applications

- 20 GiB idle memory

Baselines:

1. Overprovisioning
   (67.5 GiB soft memory usage)
2. **Static Provisioning**
   - 5GiB per app



Colocating Four Applications

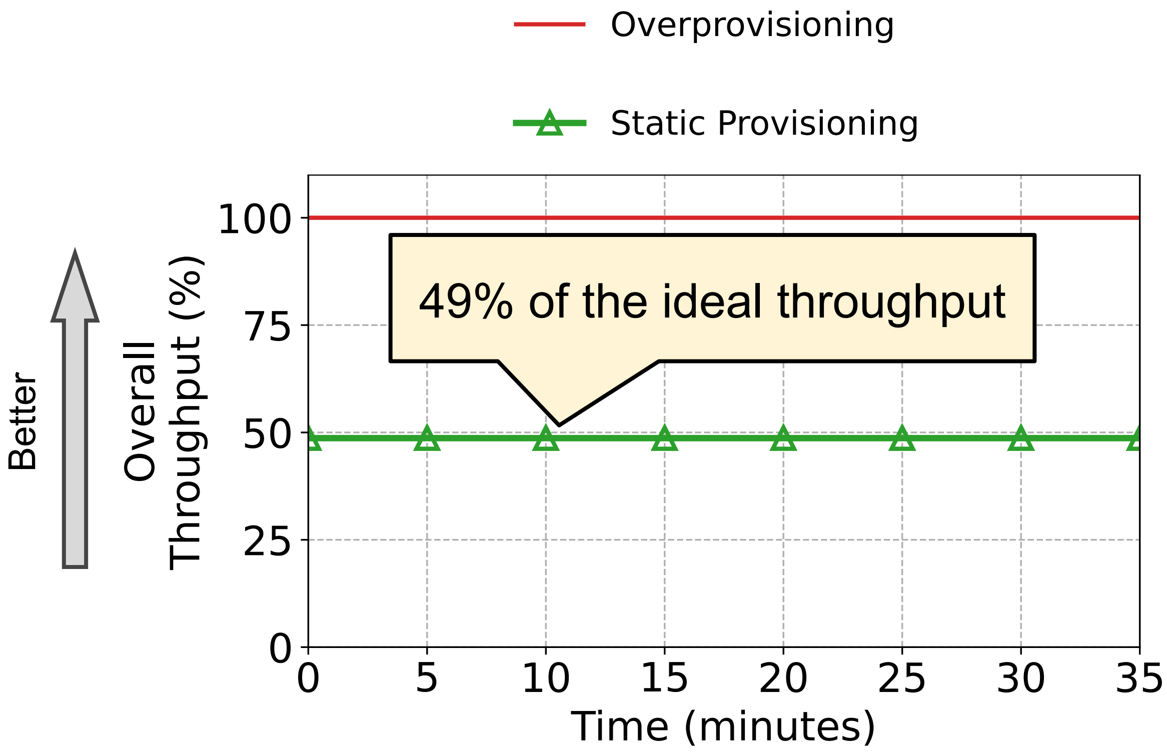
- 20 GiB idle memory

Baselines:

1. Overprovisioning

   (67.5 GiB soft memory usage)

2. **Static Provisioning**
   - 5GiB per app



49% of the ideal throughput

Colocating Four Applications

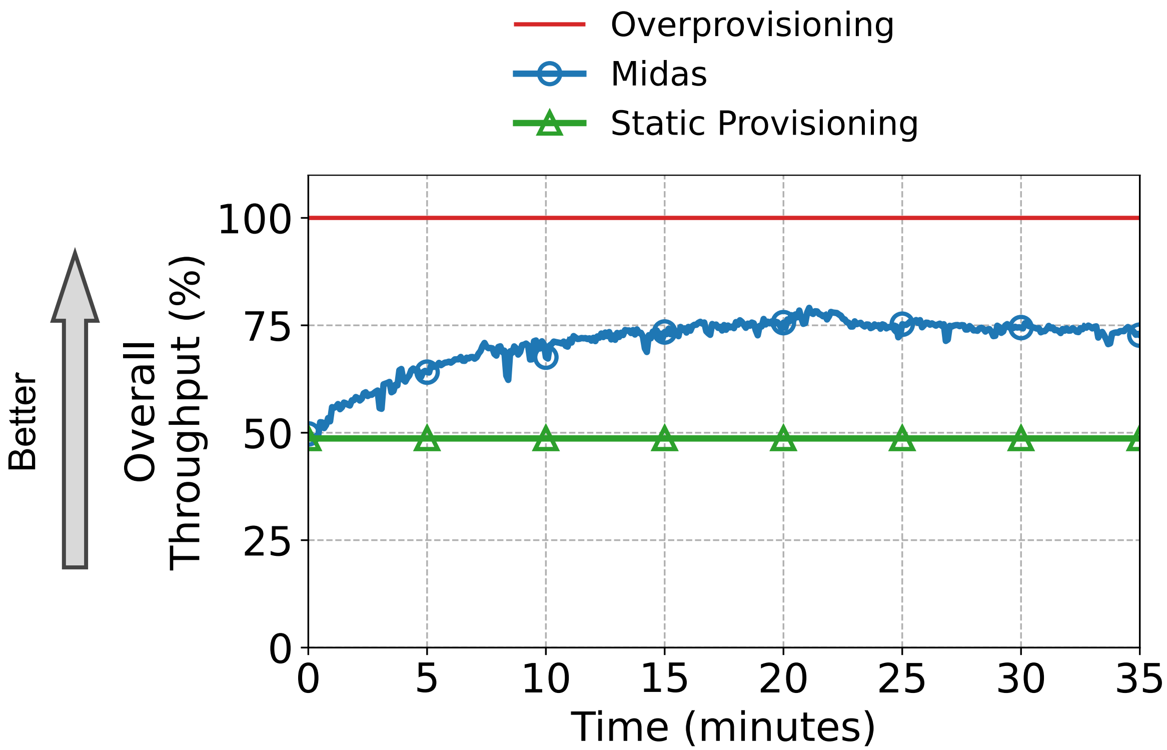
- 20 GiB idle memory

Baselines:

1. Overprovisioning
   (67.5 GiB soft memory usage)
2. Static Provisioning
   - 5GiB per app

Midas

- Initially 5GiB per app
- Dynamically coordinate



Colocating Four Applications

- 20 GiB idle memory

Baselines:

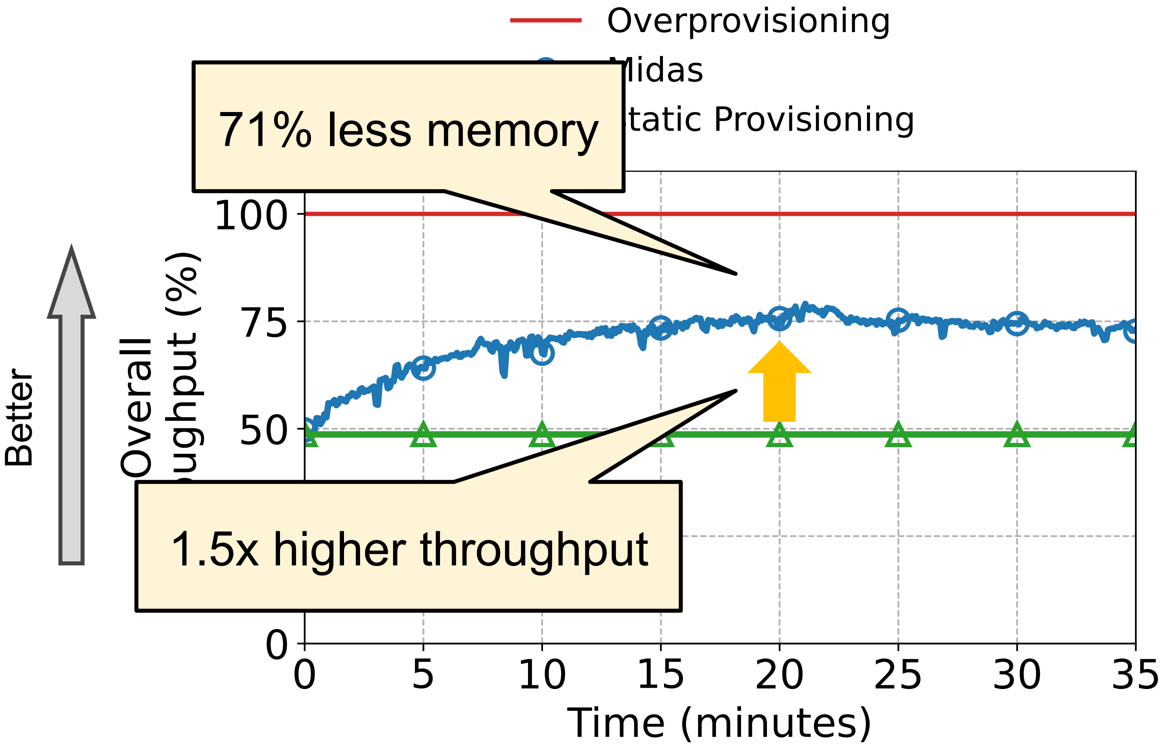
1. Overprovisioning

   (67.5 GiB soft memory usage)

2. Static Provisioning
   - 5GiB per app

Midas

- Initially 5GiB per app
- Dynamically coordinate

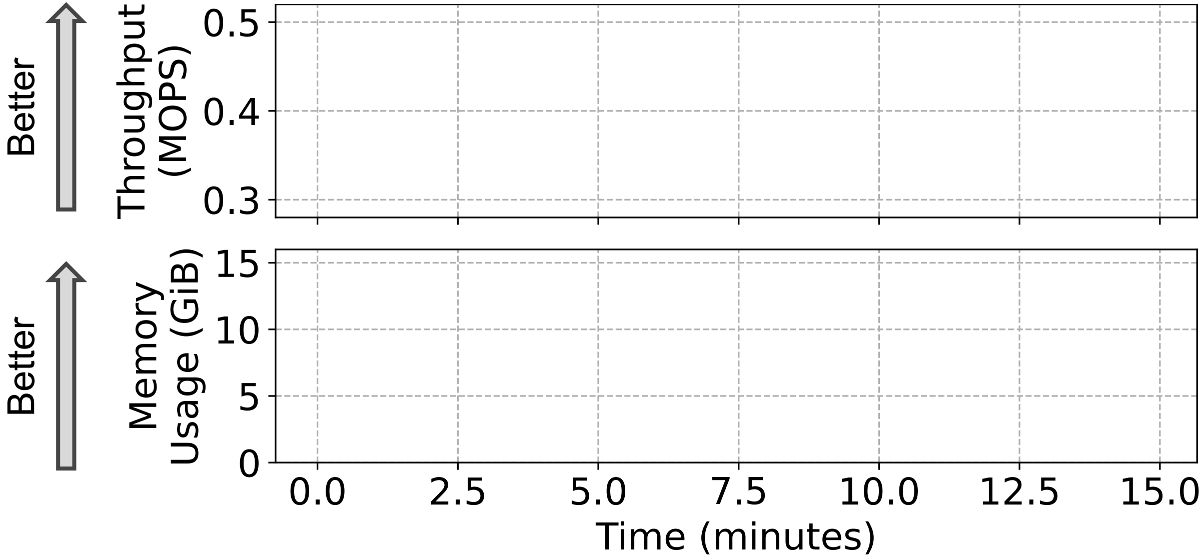


71% less memory

1.5x higher throughput

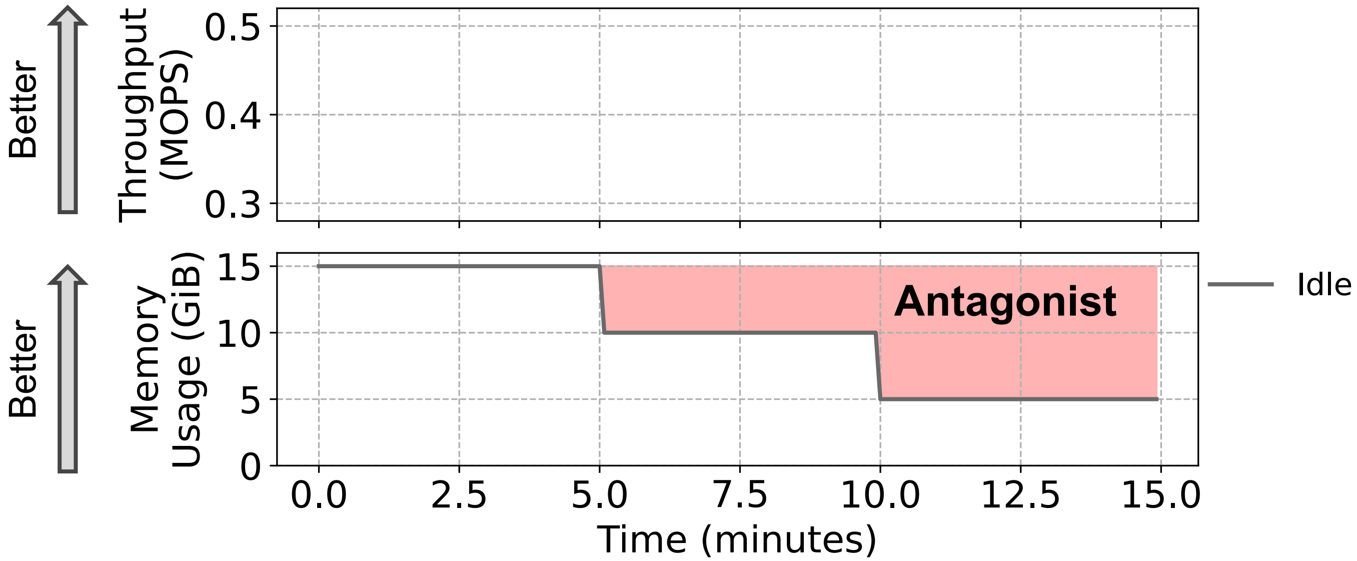
Reacting to Memory Pressure

- Run WiredTiger with 15 GiB soft memory initially



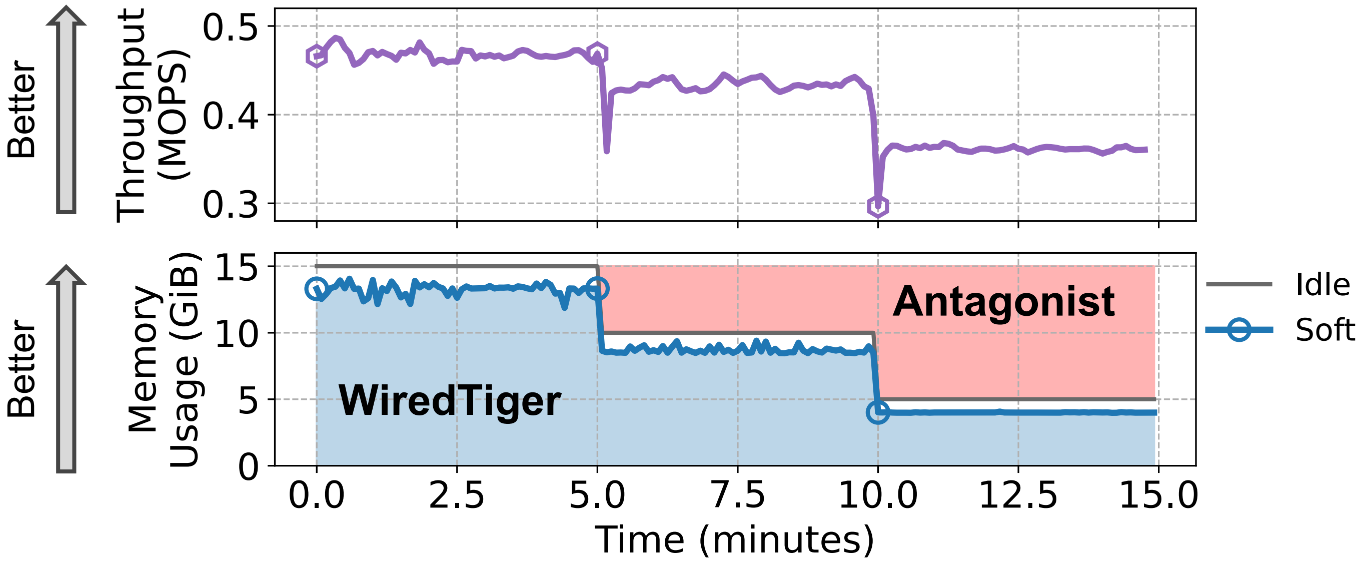
Reacting to Memory Pressure

- Run WiredTiger with 15 GiB soft memory initially
- Then launch the memory antagonist
  - **Fast** memory allocation (7 GiB/s) at t=5min and t=10min



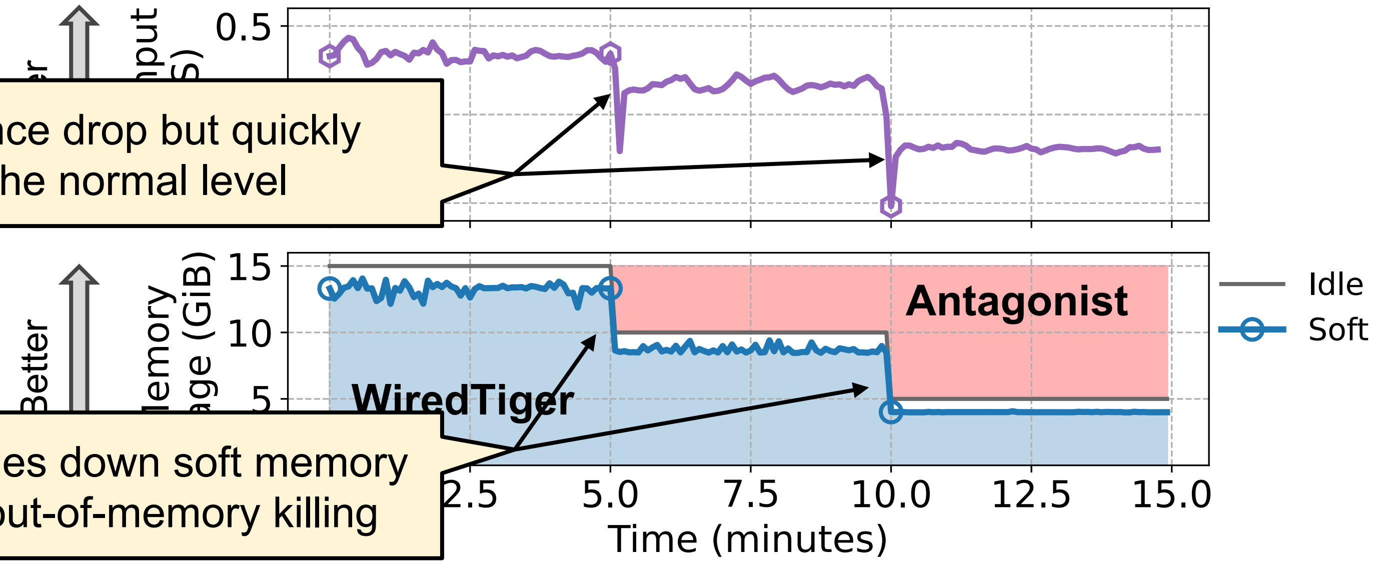
Reacting to Memory Pressure

- Run WiredTiger with 15 GiB soft memory initially
- Then launch the memory antagonist
  - **Fast** memory allocation (7 GiB/s) at t=5min and t=10min



Reacting to Memory Pressure

- Run WiredTiger with 15 GiB soft memory initially
- Then launch the memory antagonist
  - **Fast** memory allocation (7 GiB/s) at t=5min and t=10min



Brief performance drop but quickly recover to the normal level

Midas rapidly scales down soft memory usage to avoid out-of-memory killing

Conclusion

Midas enables applications to harvest idle memory for application soft state

Key designs:

1. The soft memory abstraction offering seemingly unlimited cache space
2. A runtime that manages soft state in available idle memory
3. OS kernel support that quickly reclaims memory under pressure

https://github.com/uclasystem/midas

Thank You!