# Building Efficient OS X Apps

## Advanced Topics in Resource Management

Session 704

**Anthony Chivetta**
Performance Engineer

# Introduction

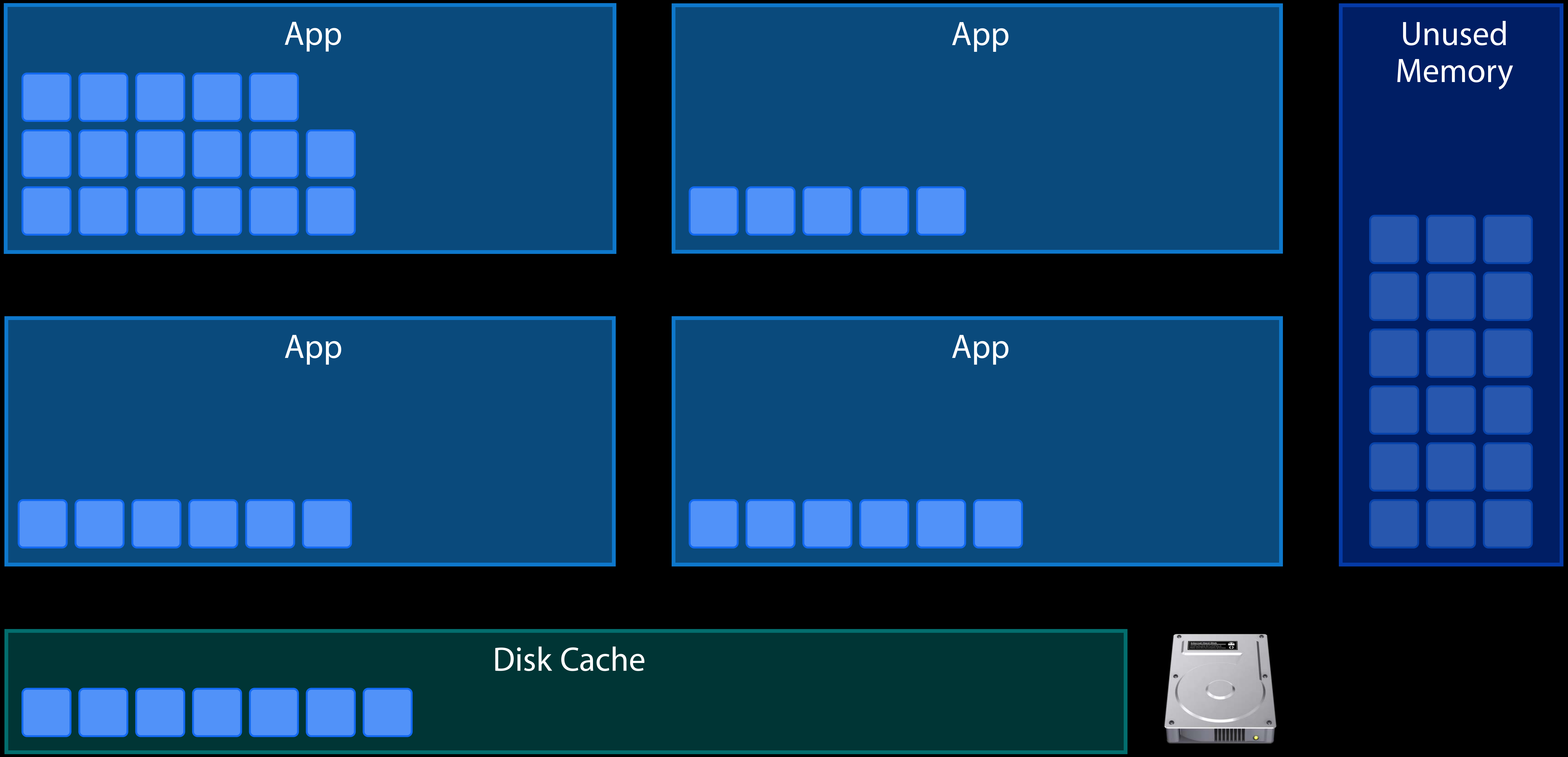
- Use shared resources efficiently
- Apps affect each other’s performance
- Create great user experience

# What You Will Learn

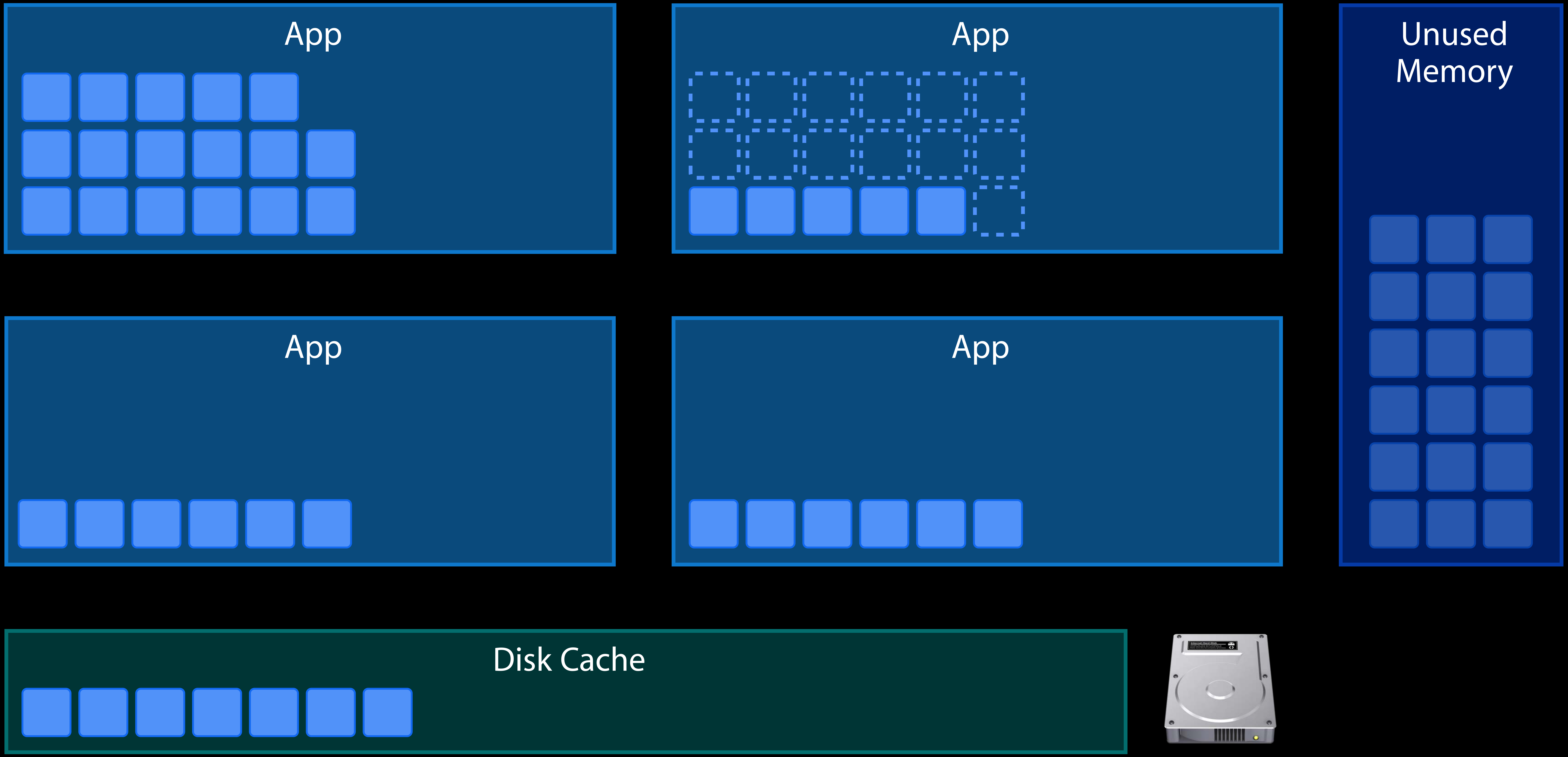
- How to reduce memory footprint
- How to optimize disk accesses
- How to do background work

# Memory

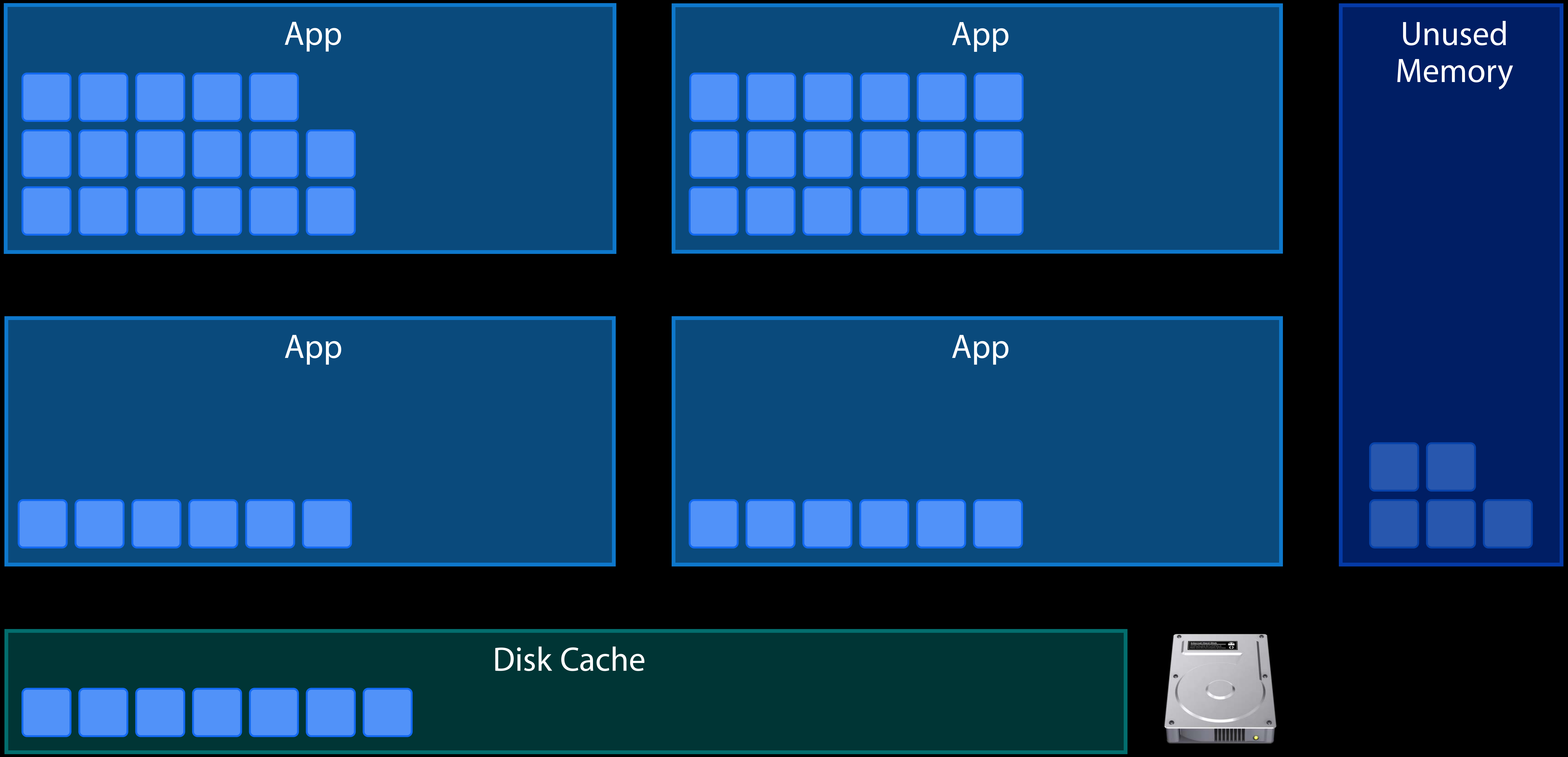
# Why Is Memory Use Important?



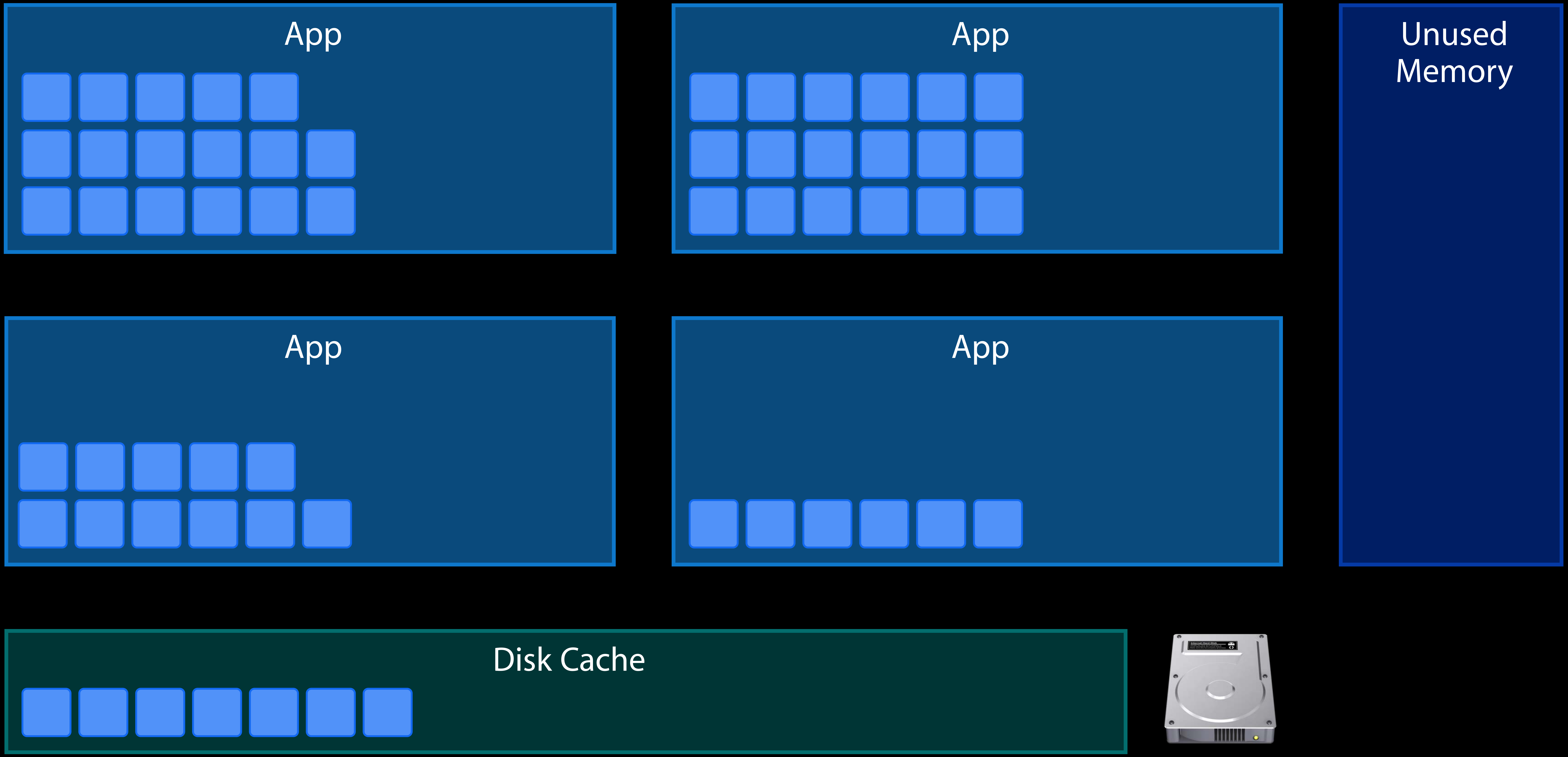
# Why Is Memory Use Important?



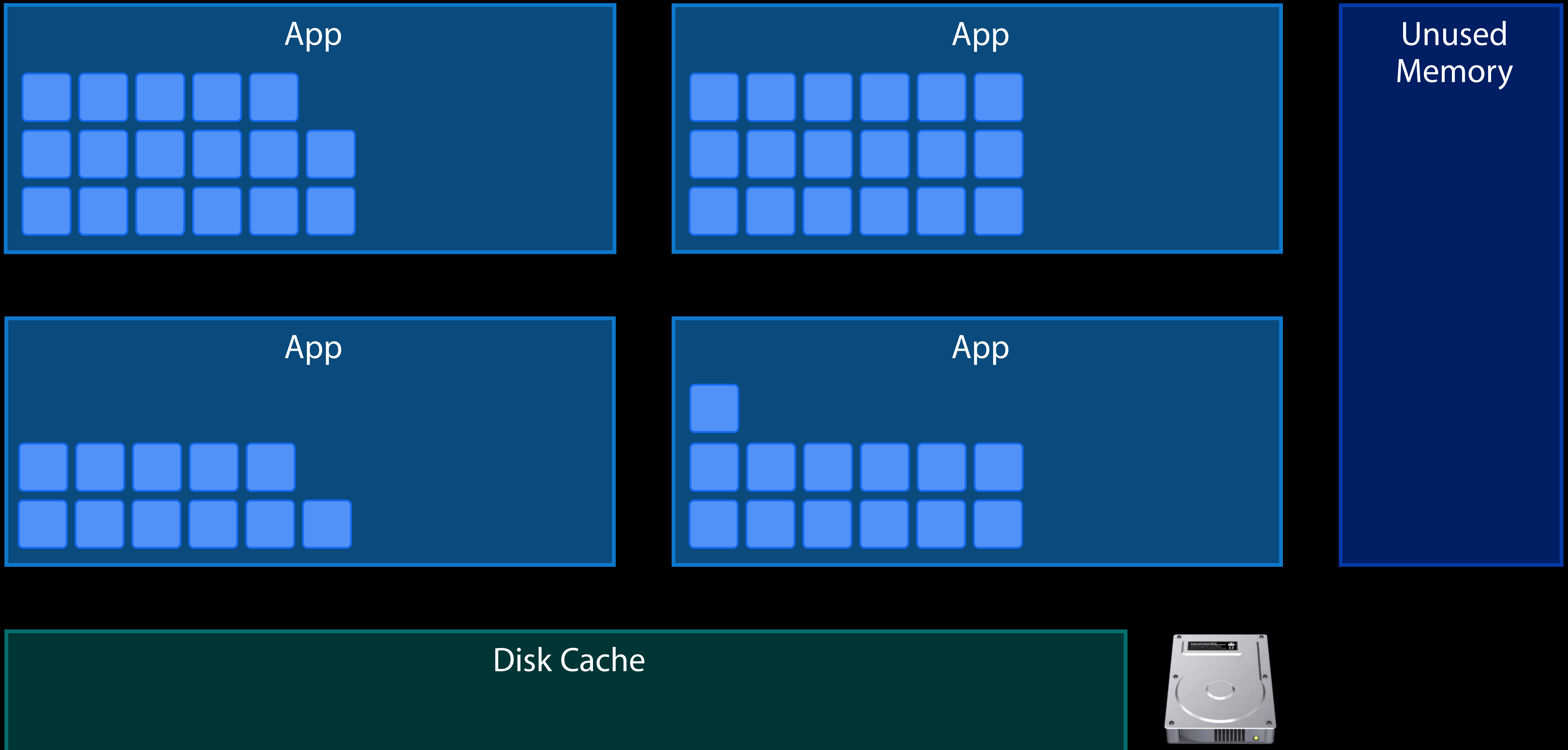
# Why Is Memory Use Important?



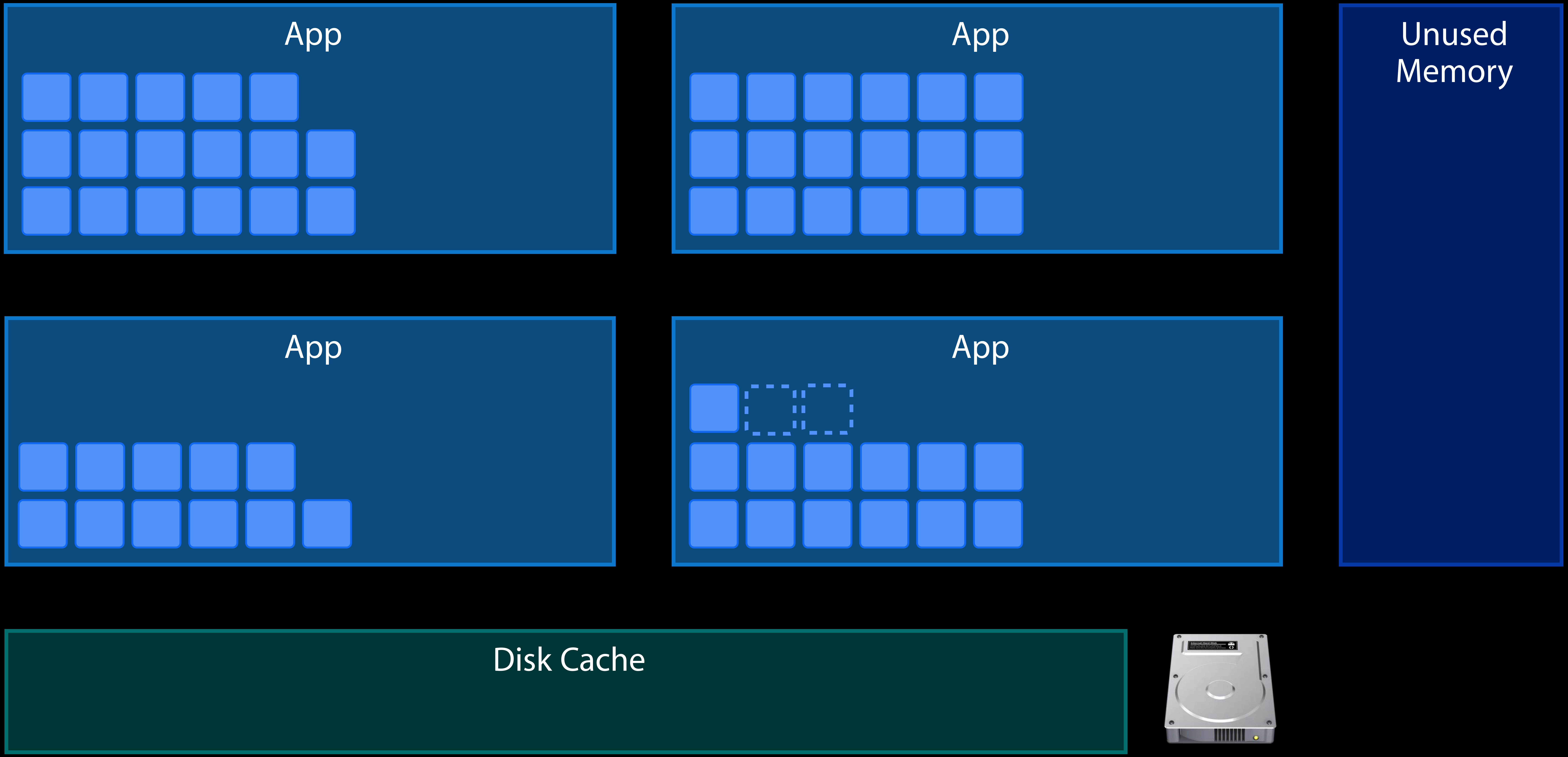
# Why Is Memory Use Important?



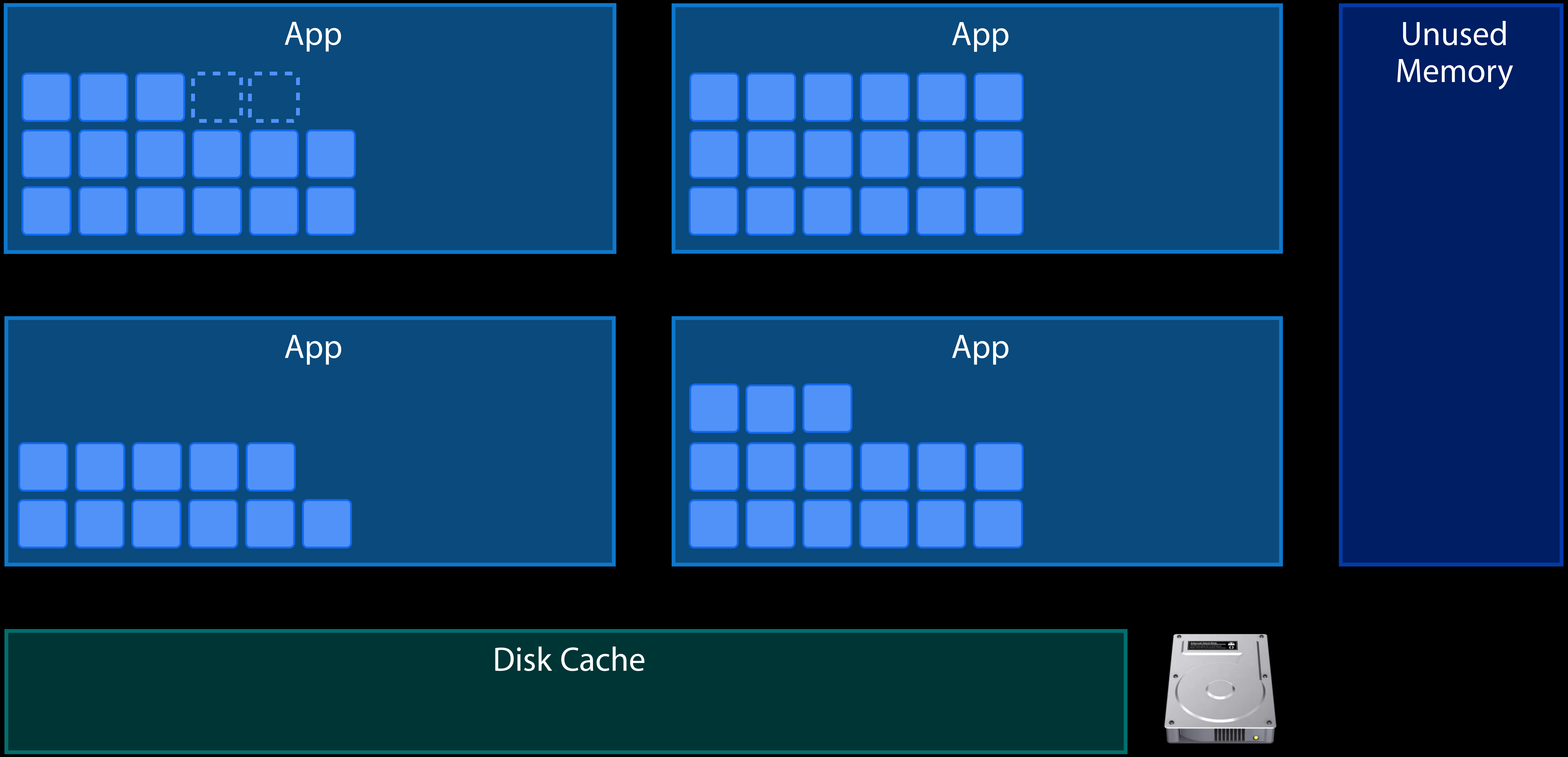
# Why Is Memory Use Important?



# Why Is Memory Use Important?



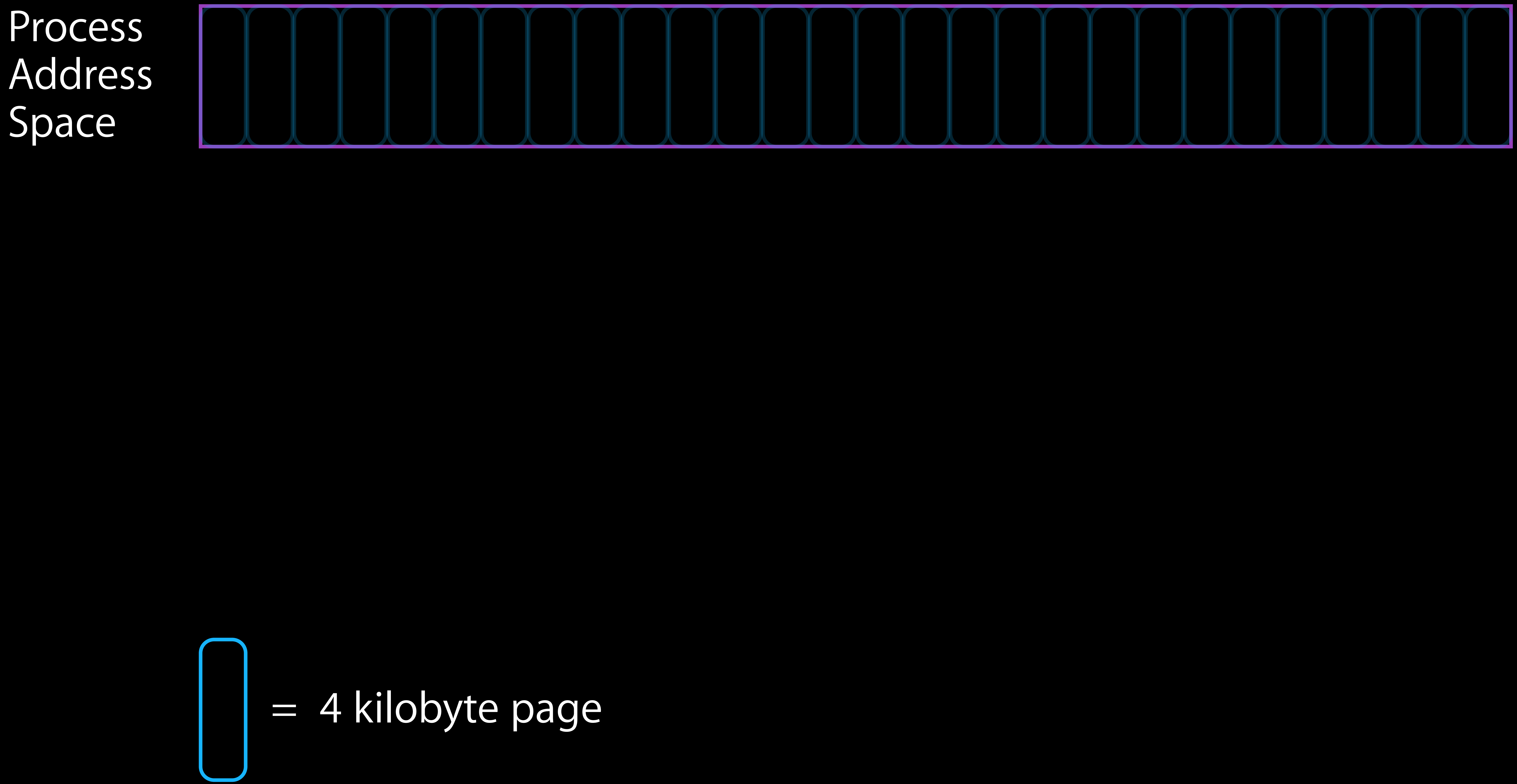
# Why Is Memory Use Important?



# Virtual Memory

Process
Address
Space

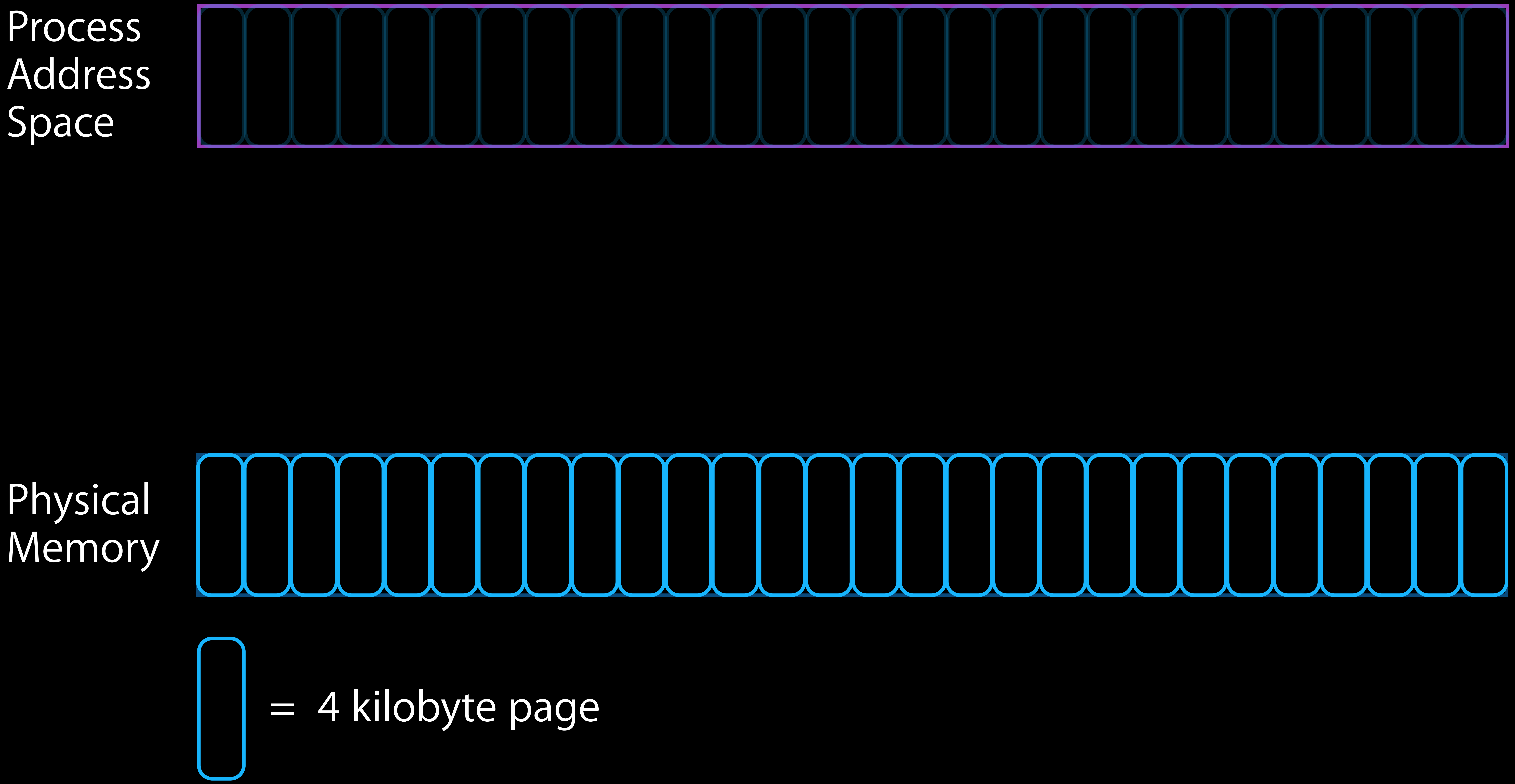
# Virtual Memory



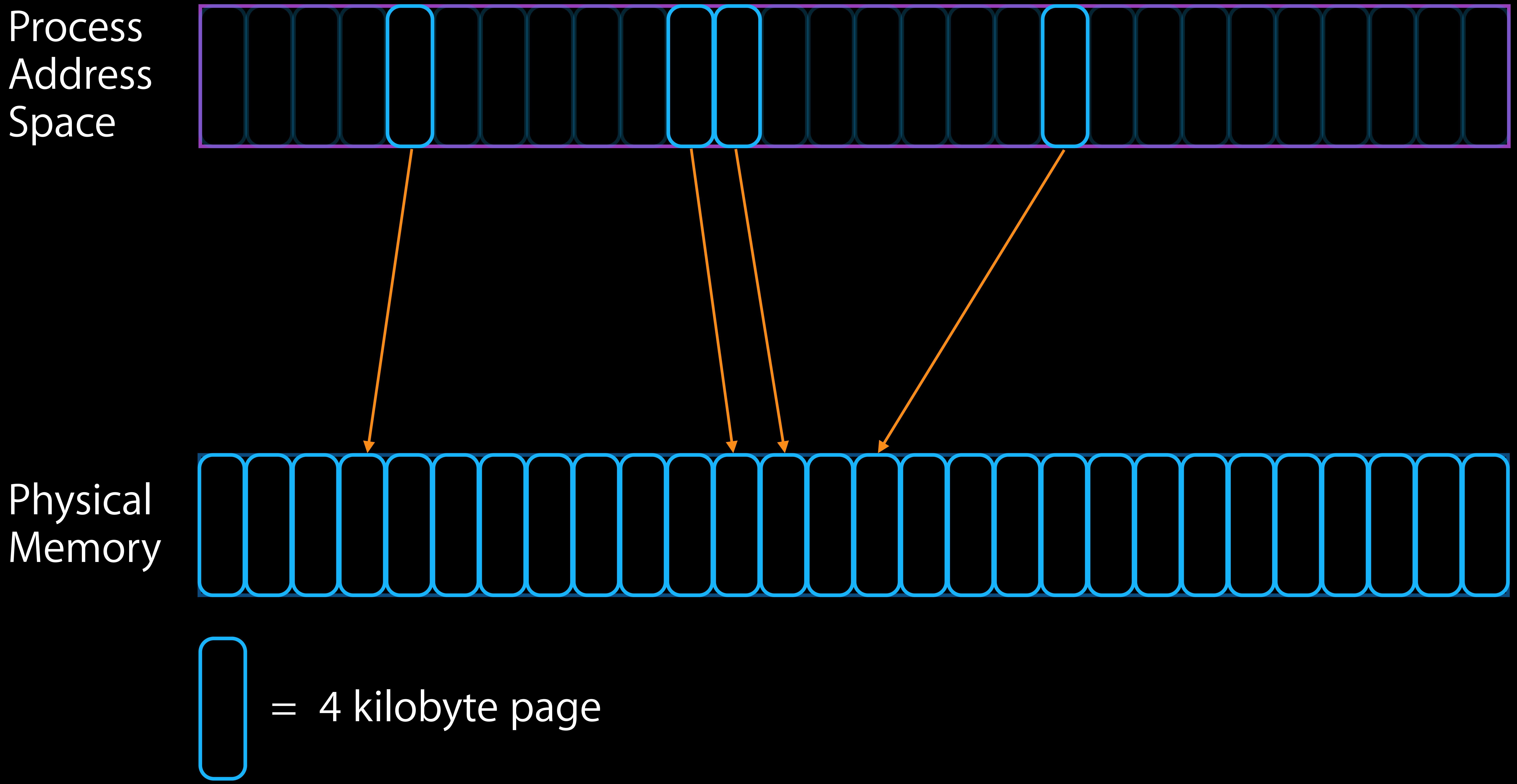
Process Address Space

▯ = 4 kilobyte page

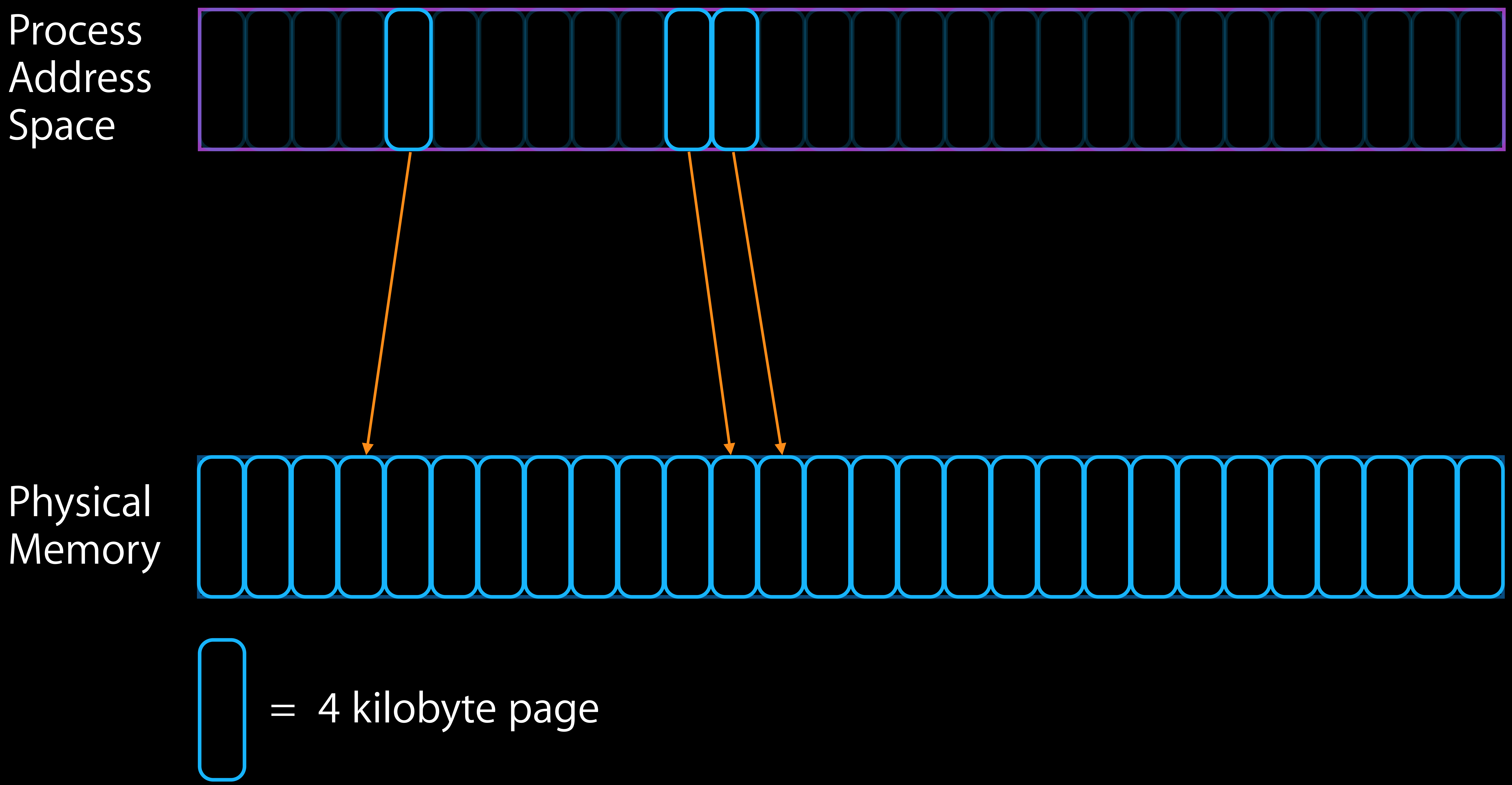
# Virtual Memory



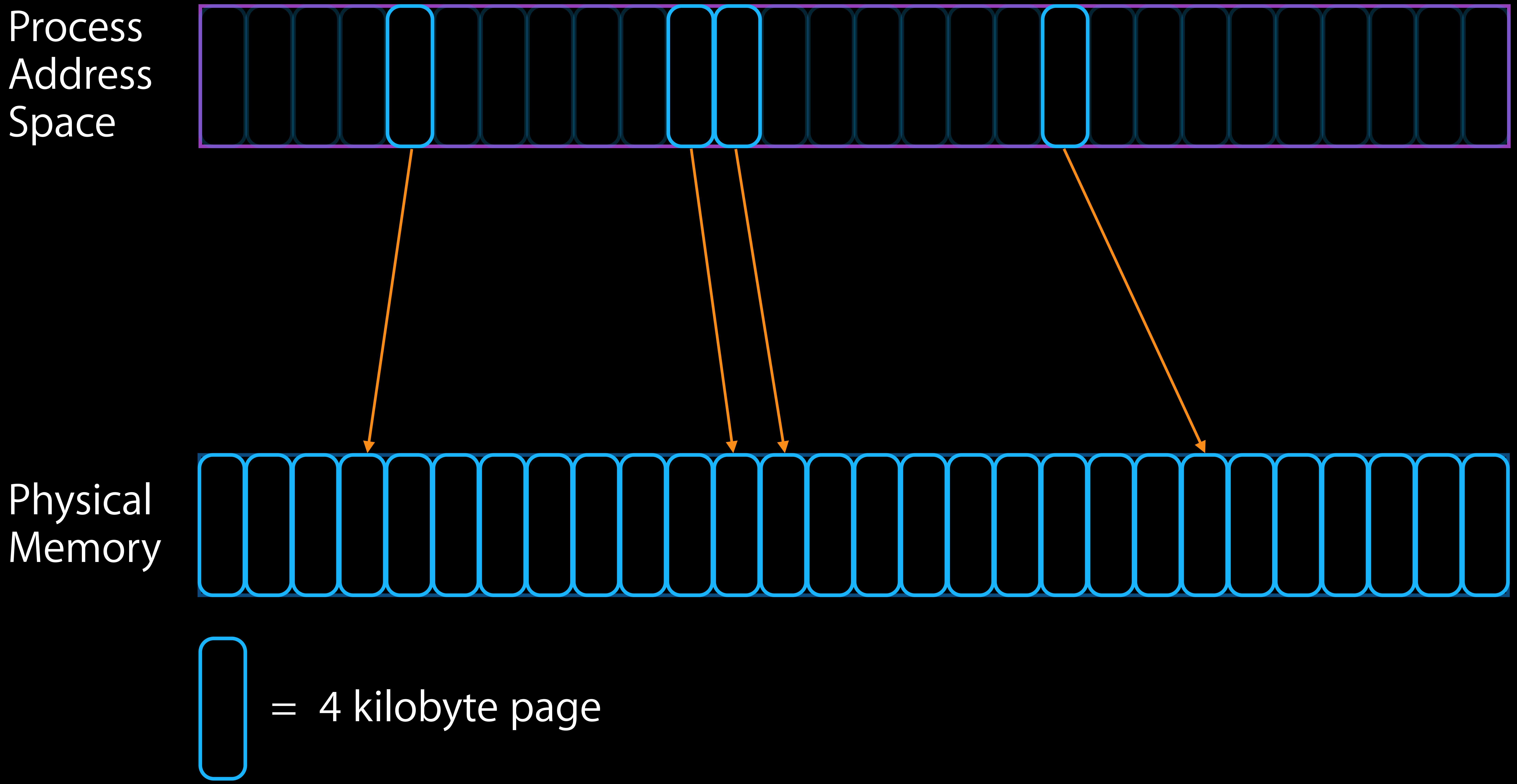
# Virtual Memory



# Virtual Memory



# Virtual Memory



# Lower Your Memory Footprint

- Reduces chance your memory is swapped
- More memory is quickly available when needed
- Improves overall system performance

# Profile and Reduce Memory Use

Allocations

- Profile objects allocated by you app
- Helps find areas to focus optimization efforts

Leaks

- Look for leaked objects
- Analyze retain cycles

**Fixing Memory Issues**

Nob Hill
Thursday 2:00PM

# Automate Memory Testing

- Integrate memory metrics with your regular testing
- View increases in allocated objects with suspicion
- Immediately fix leaks to prevent engineering debt

# Automated Allocations Profiling

- Use the `heap` command-line tool

# Automated Allocations Profiling

- Use the heap command-line tool

```
$ heap MyLeakyApp
```

# Automated Allocations Profiling

- Use the heap command-line tool

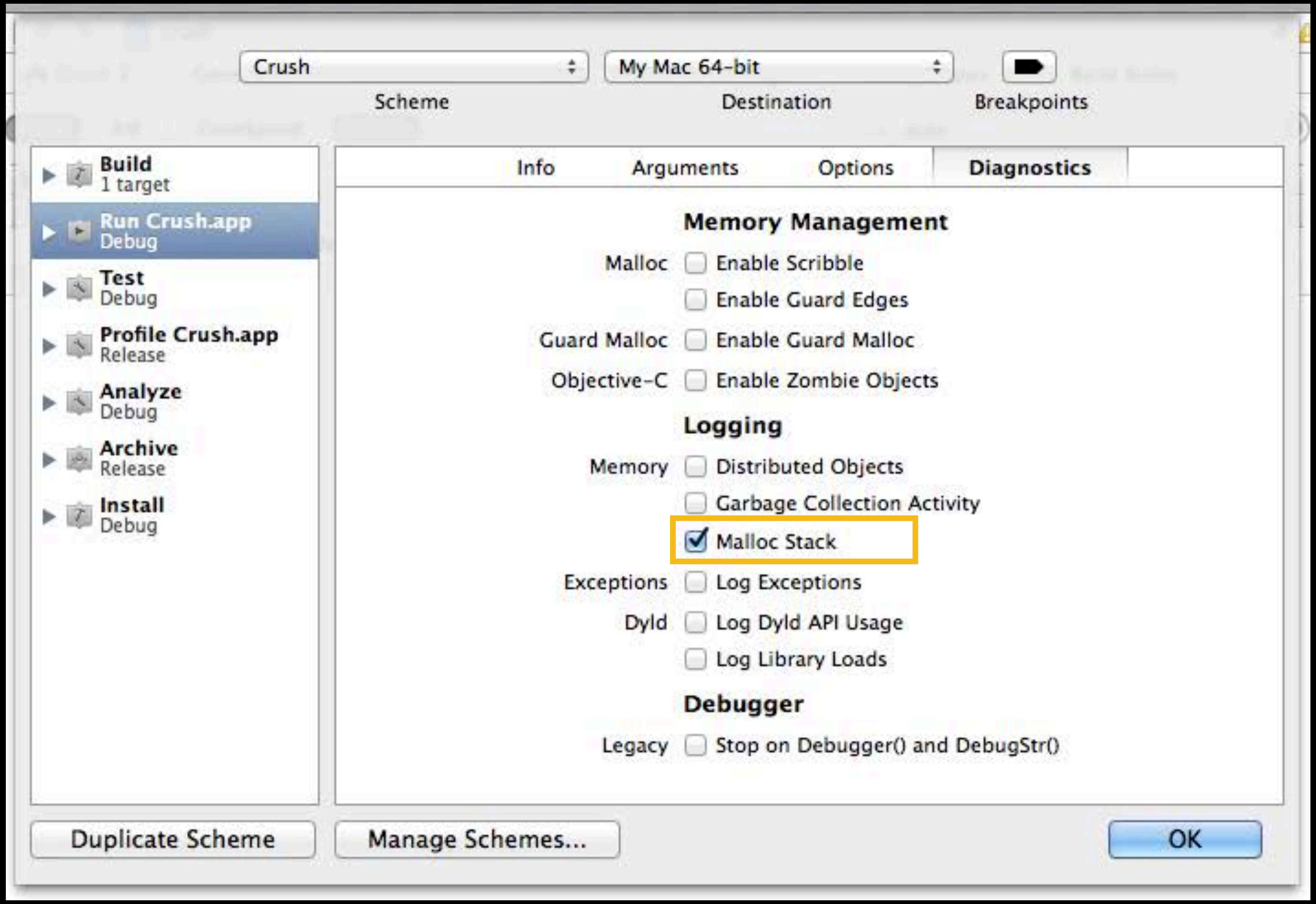
```
$ heap MyLeakyApp

  COUNT     BYTES       AVG   CLASS_NAME          TYPE    BINARY
  =====     =====       ===   ==========          ====    ======
   7063    950160     134.5   non-object
   5081    234192      46.1   __NSCFString        ObjC    CoreFoundation
   1125     72000      64.0   __NSCFDictionary    ObjC    CoreFoundation
    197      9456      48.0   __NSArrayM          ObjC    CoreFoundation
    186      9680      52.0   __NSMallocBlock__   ObjC    <unknown>
    164      5248      32.0   __NSCFNumber        ObjC    CoreFoundation
    130     12480      96.0   NSMenuItem          ObjC    AppKit
     96      6144      64.0   NSURL               ObjC    CoreFoundation
```

# Automate Leak Detection

- Use the `leaks` command-line tool

# Automate Leak Detection

- Use the `leaks` command-line tool



Scheme: Crush
Destination: My Mac 64-bit

- Build 1 target
- Run Crush.app Debug
- Test Debug
- Profile Crush.app Release
- Analyze Debug
- Archive Release
- Install Debug

**Memory Management**

Malloc ☐ Enable Scribble
☐ Enable Guard Edges
Guard Malloc ☐ Enable Guard Malloc
Objective-C ☐ Enable Zombie Objects

**Logging**

Memory ☐ Distributed Objects
☐ Garbage Collection Activity
☑ Malloc Stack
Exceptions ☐ Log Exceptions
Dyld ☐ Log Dyld API Usage
☐ Log Library Loads

**Debugger**

Legacy ☐ Stop on Debugger() and DebugStr()

```
MallocStackLoggging=1
```

# Automate Leak Detection

- Use the `leaks` command-line tool

```
$ leaks MyLeakyApp
```

# Automate Leak Detection

- Use the `leaks` command-line tool

```
$ leaks MyLeakyApp
leaks Report Version:  2.0
Process 60641: 11227 nodes malloced for 1150 KB
Process 60641: 3 leaks for 96 total leaked bytes.
```

# Automate Leak Detection

- Use the `leaks` command-line tool

```
$ leaks MyLeakyApp
leaks Report Version:  2.0
Process 60641: 11227 nodes malloced for 1150 KB
Process 60641: 3 leaks for 96 total leaked bytes.
Leak: 0x7f9ef172ebd0  size=16  zone: DefaultMallocZone_0x10b68e000
MyLeakedClass  ObjC  MyLeakyApp
```

# Automate Leak Detection

- Use the `leaks` command-line tool

```
$ leaks MyLeakyApp
leaks Report Version:  2.0
Process 60641: 11227 nodes malloced for 1150 KB
Process 60641: 3 leaks for 96 total leaked bytes.
Leak: 0x7f9ef172ebd0  size=16  zone: DefaultMallocZone_0x10b68e000
MyLeakedClass  ObjC  MyLeakyApp
   Call stack: [thread 0x7fff777ce310]: | 0x1 | start | main main.m:13 |
NSApplicationMain | -[NSApplication run] | <snip> | -
[AppDelegate applicationDidFinishLaunching:] AppDelegate.m:16 | +
[NSObject allocWithZone:] | class_createInstance | calloc |
malloc_zone_calloc
```

# Avoid Duplicate Objects

- `stringdups` finds duplicate objects
    - Examines C strings, NSString, NSDate, and more

# Avoid Duplicate Objects

- `stringdups` finds duplicate objects
  - Examines C strings, NSString, NSDate, and more

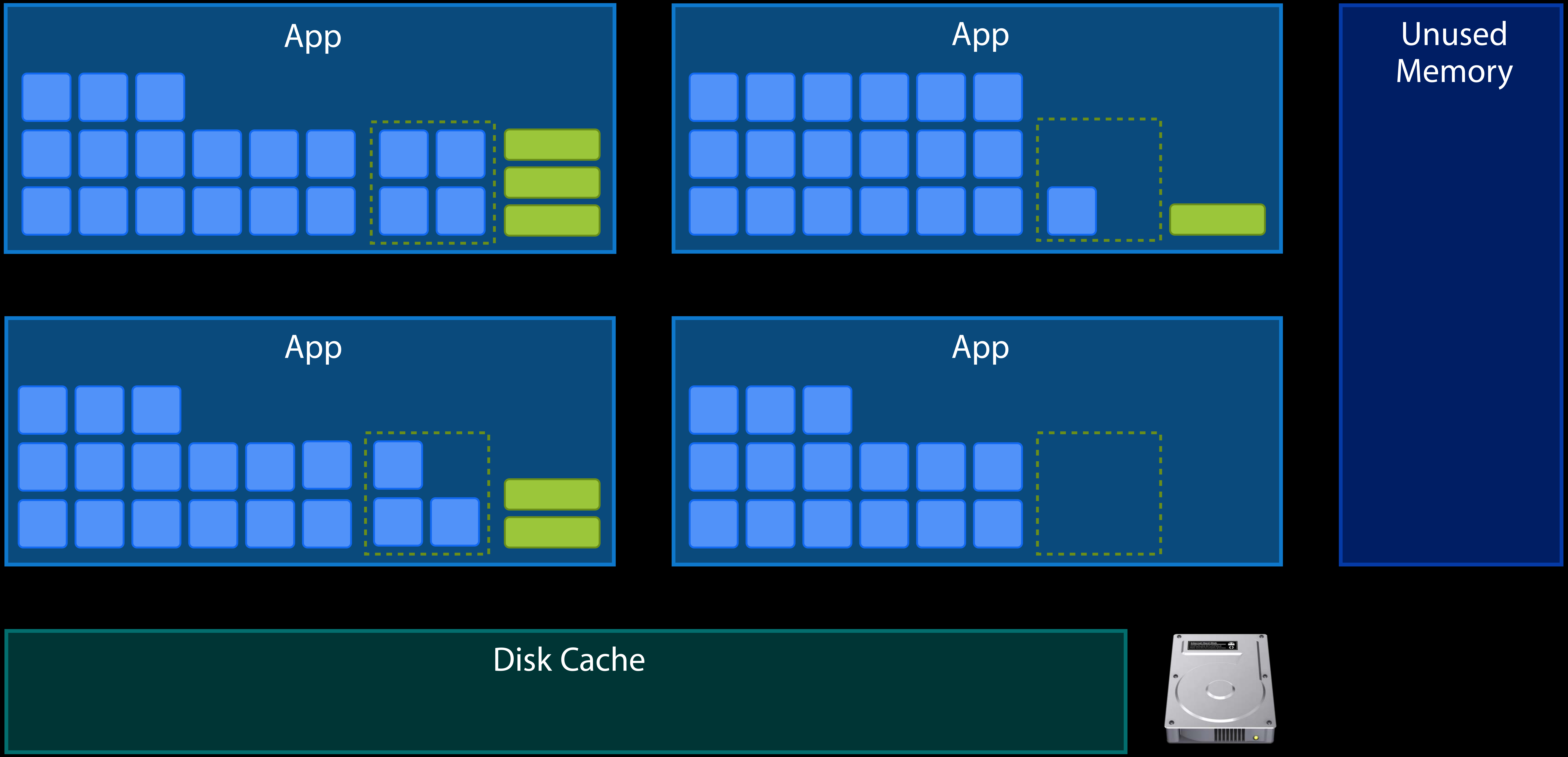
```
$ stringdups -nostacks <pid>
    COUNT     BYTES   AVERAGE   CONTENT
    =====     =====   =======   =======
        2        96      48.0   __NSCFString  "This is a duplicate"
```

# Avoid Duplicate Objects

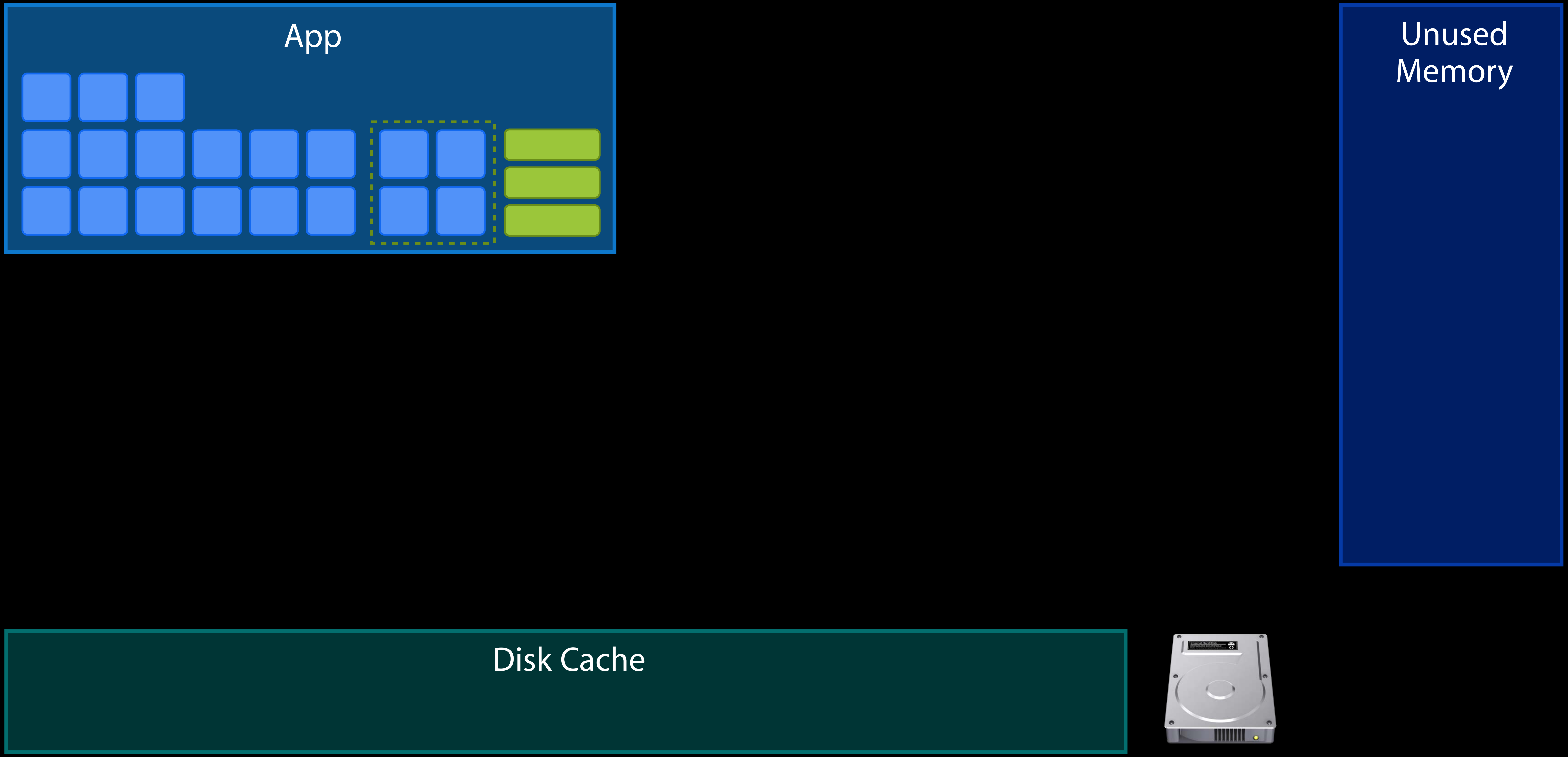
- stringdups finds duplicate objects
    - Examines C strings, NSString, NSDate, and more

```
$ stringdups -nostacks <pid>
    COUNT     BYTES   AVERAGE   CONTENT
    =====     =====   =======   =======
        2        96      48.0   __NSCFString  "This is a duplicate"

$ stringdups -callTrees <pid>
Instances: 2   Total bytes: 96   Average bytes: 48.0
              __NSCFString  "This is a duplicate"
Call tree:
  2 (96) << TOTAL >>
    2 (96) Thread_777ce311
...
      1 (48) -[MyLeakedClass init]  (in leaks) + 70  MyLeakedClass.m:14
        1 (48) +[NSString stringWithUTF8String:] (in Foundation) + 131
```

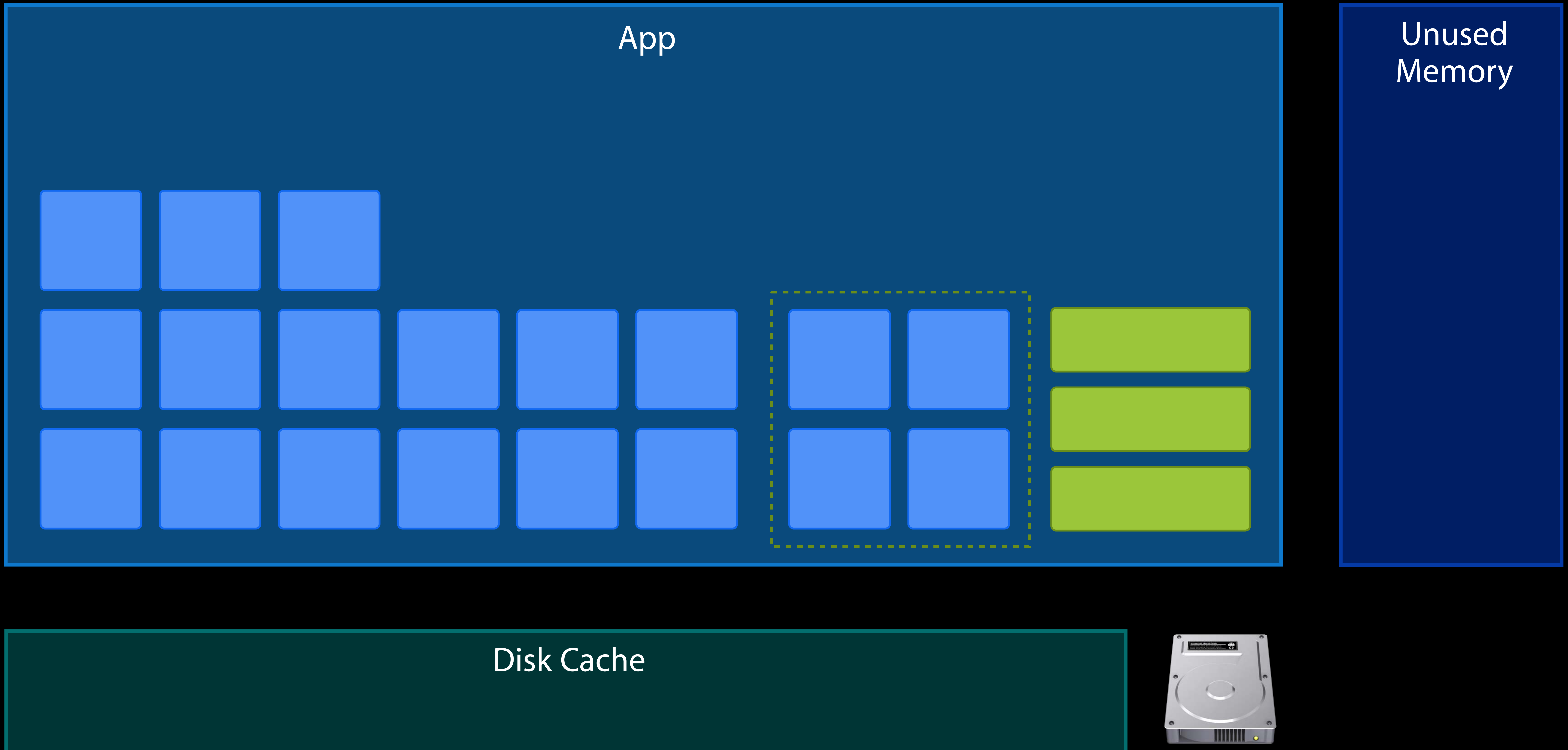
# Memory Pressure



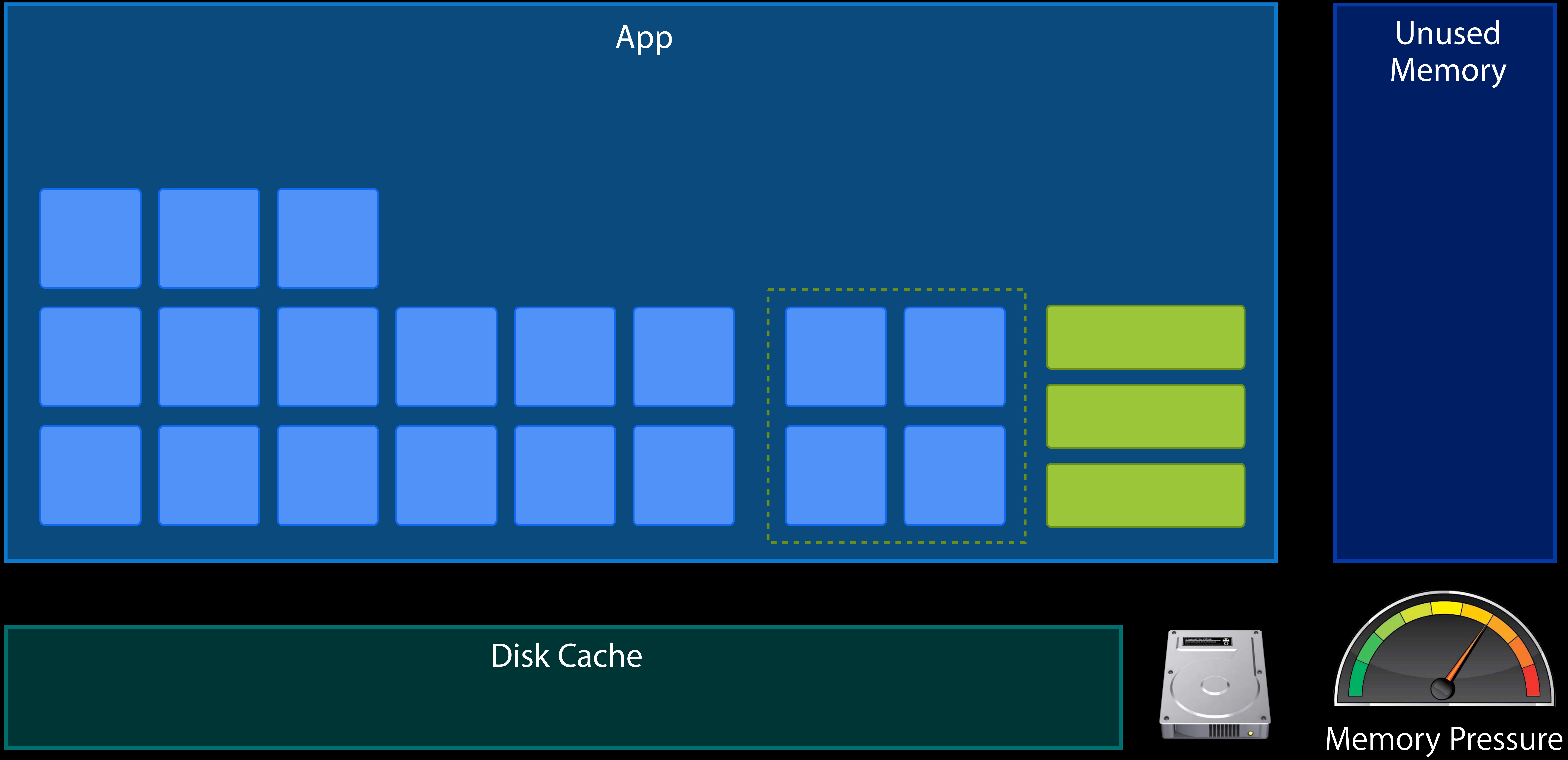
# Memory Pressure



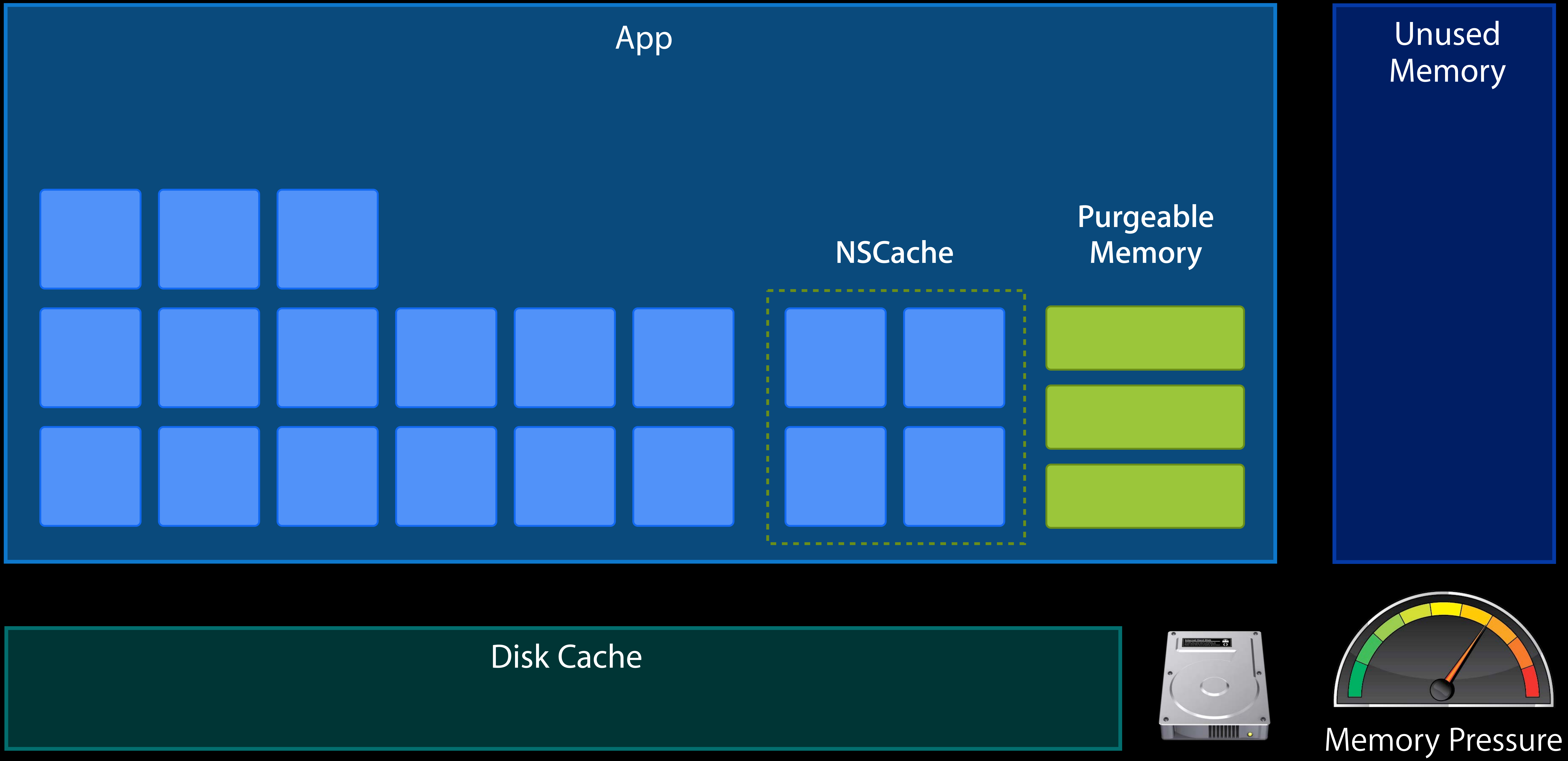
# Memory Pressure



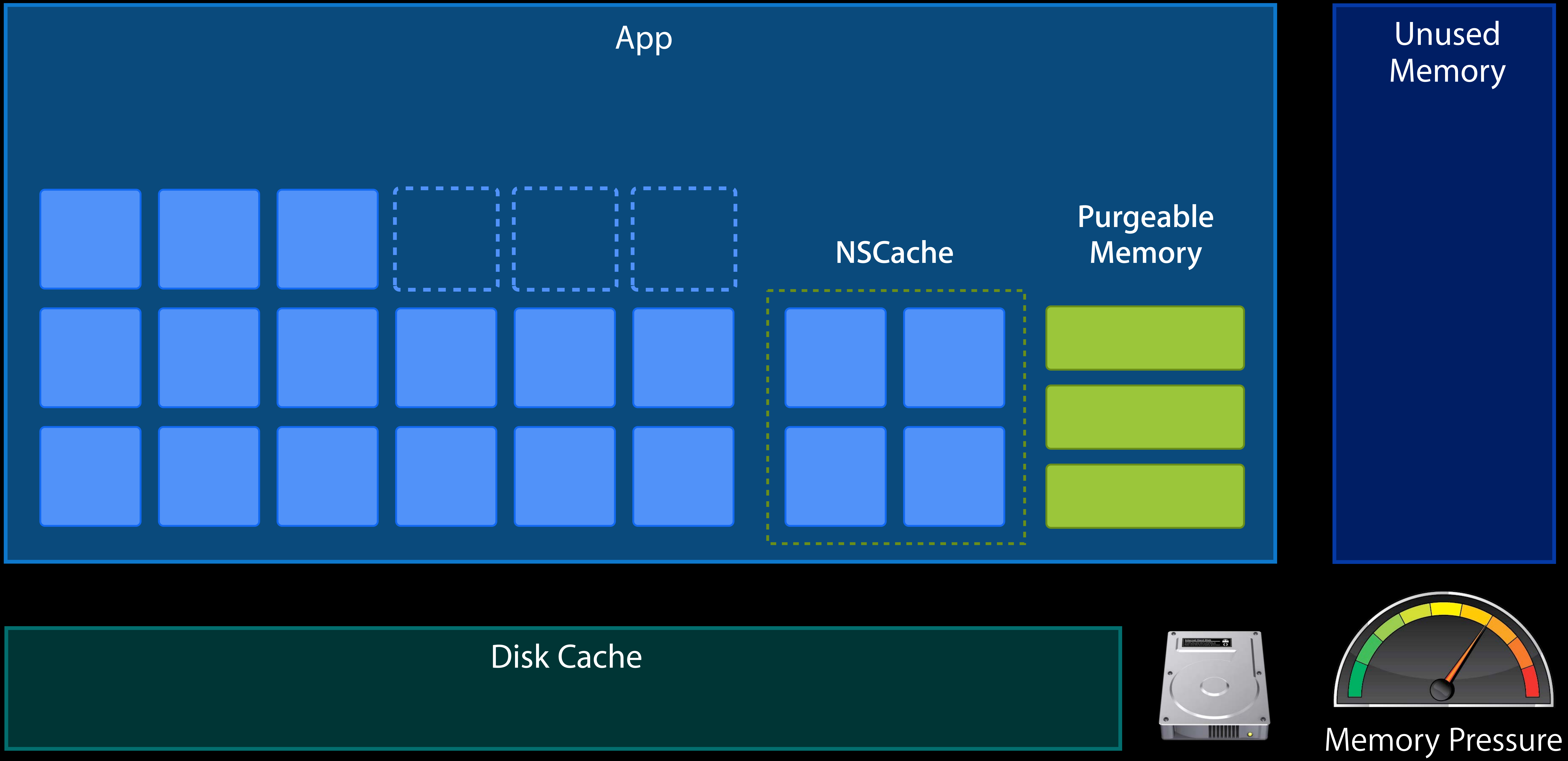
# Memory Pressure



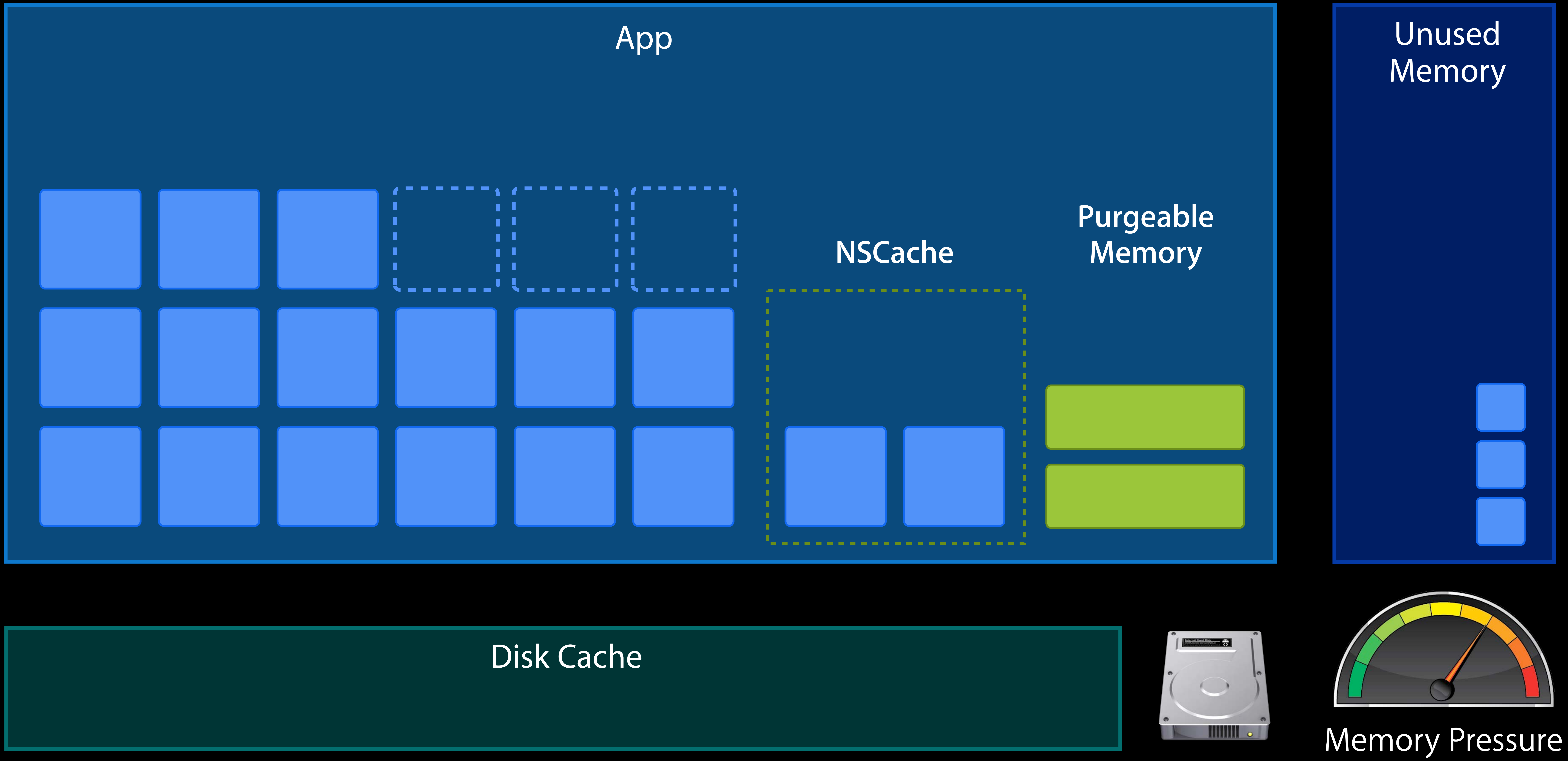
# Memory Pressure



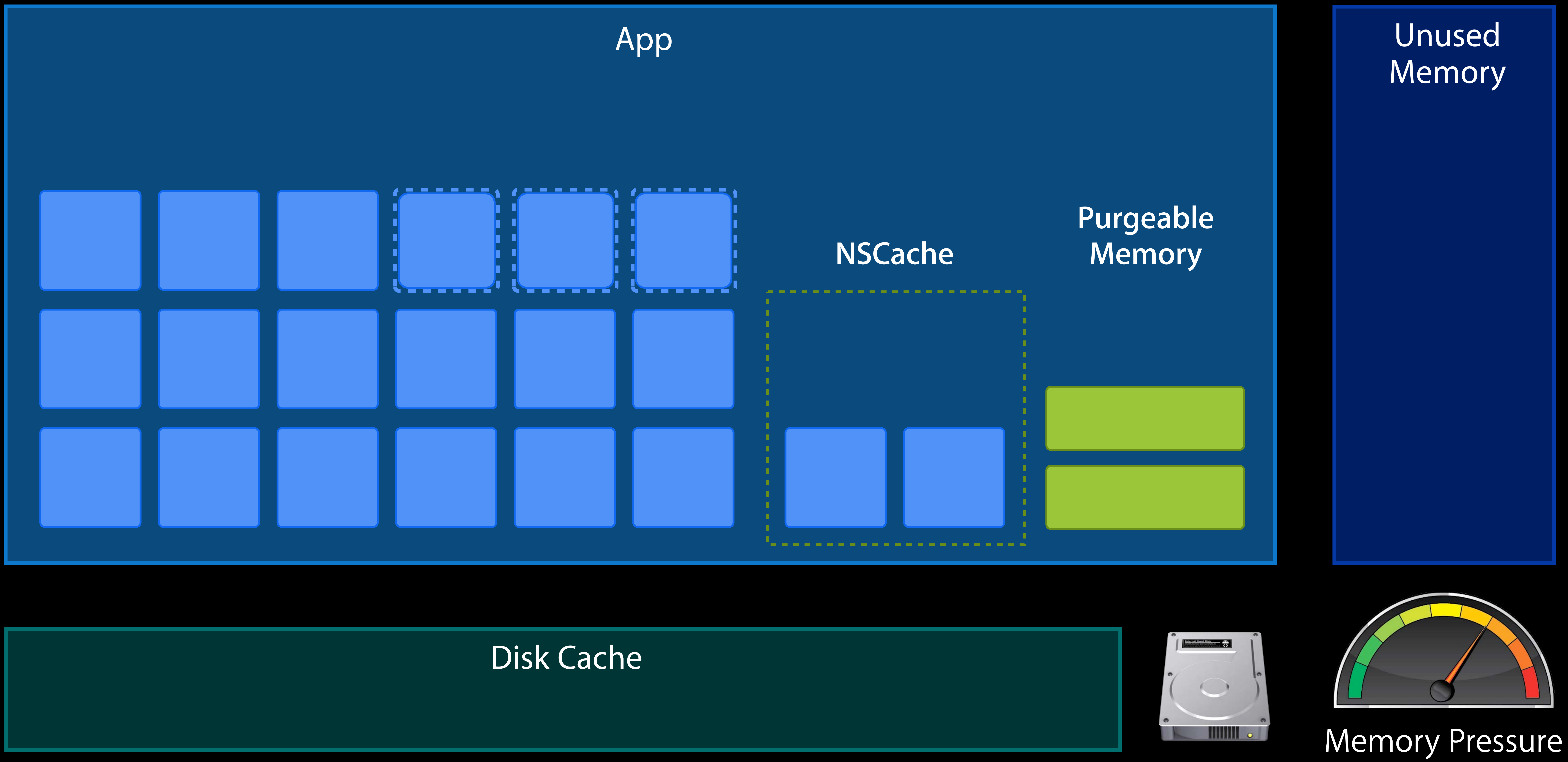
# Memory Pressure



# Memory Pressure

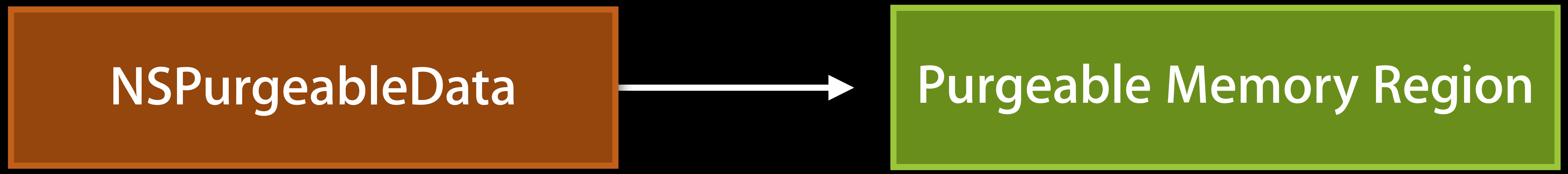


# Memory Pressure



# NSPurgeableData

- Contents discarded under memory pressure



# NSPurgeableData

- Contents discarded under memory pressure

# NSPurgeableData

- Contents discarded under memory pressure

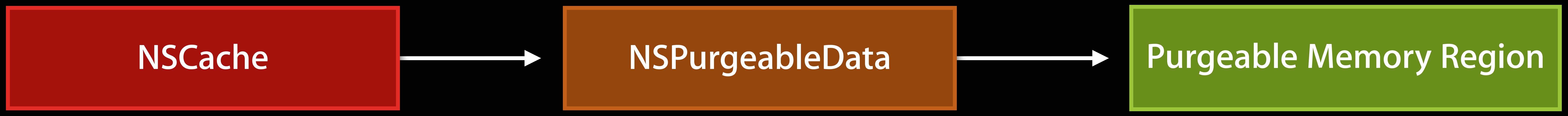
```
data = [[NSPurgeableData alloc] initWithBytes:bytes length:DATA_SIZE];
[data endContentAccess];

/* some time later */

if ([data beginContentAccess] == NO){
    /* regenerate data */
    data = [[NSPurgeableData alloc] initWithBytes:bytes length:DATA_SIZE];
}

/* use data */

[data endContentAccess];
```

# NSCache

- Like NSMutableDictionary, but thread-safe
- Automatically evicts contents on memory pressure
  - Releases reference on object
- Least recently used eviction
  - Contents will eventually be evicted

# NSPurgeableData with NSCache

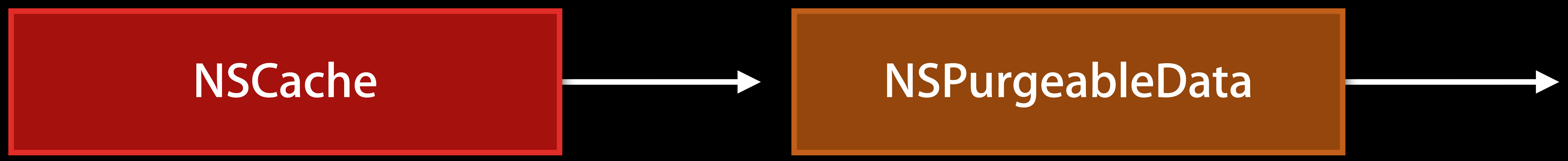
- NSCache has extra behavior for NSPurgeableData objects
  - Automatically evicted when their contents are purged



NSCache → NSPurgeableData → Purgeable Memory Region

# NSPurgeableData with NSCache

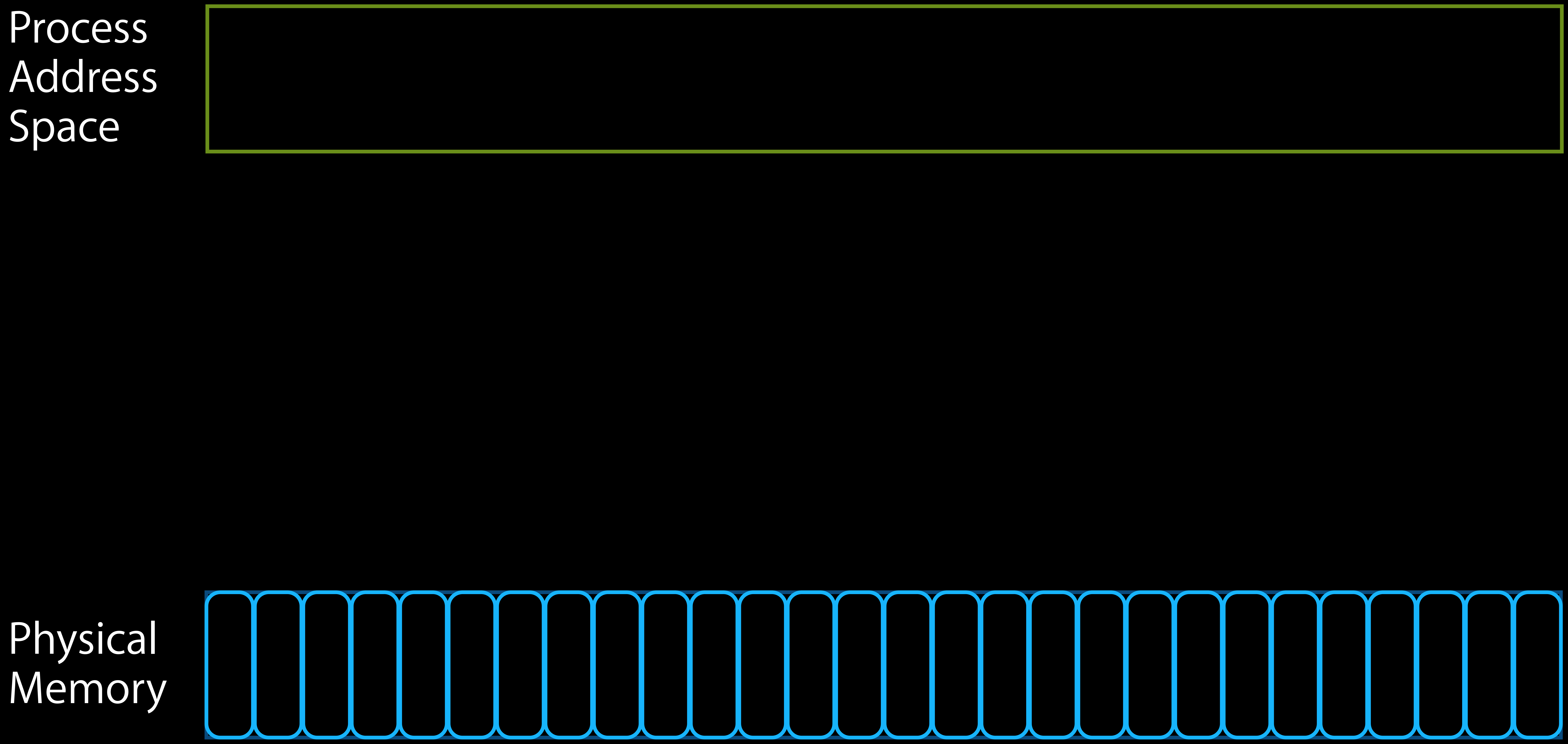
- NSCache has extra behavior for NSPurgeableData objects
  - Automatically evicted when their contents are purged



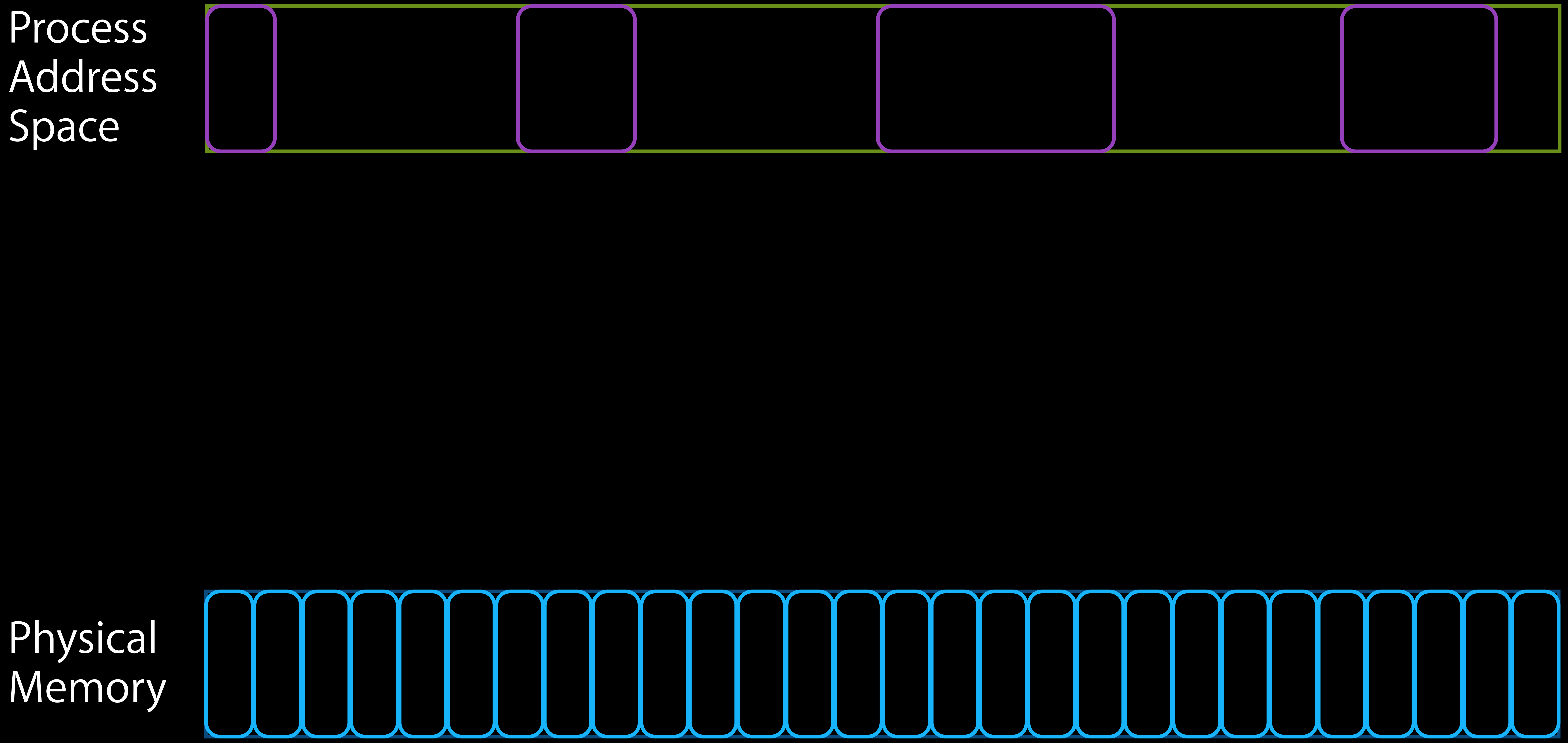
# NSPurgeableData with NSCache

- NSCache has extra behavior for NSPurgeableData objects
  - Automatically evicted when their contents are purged

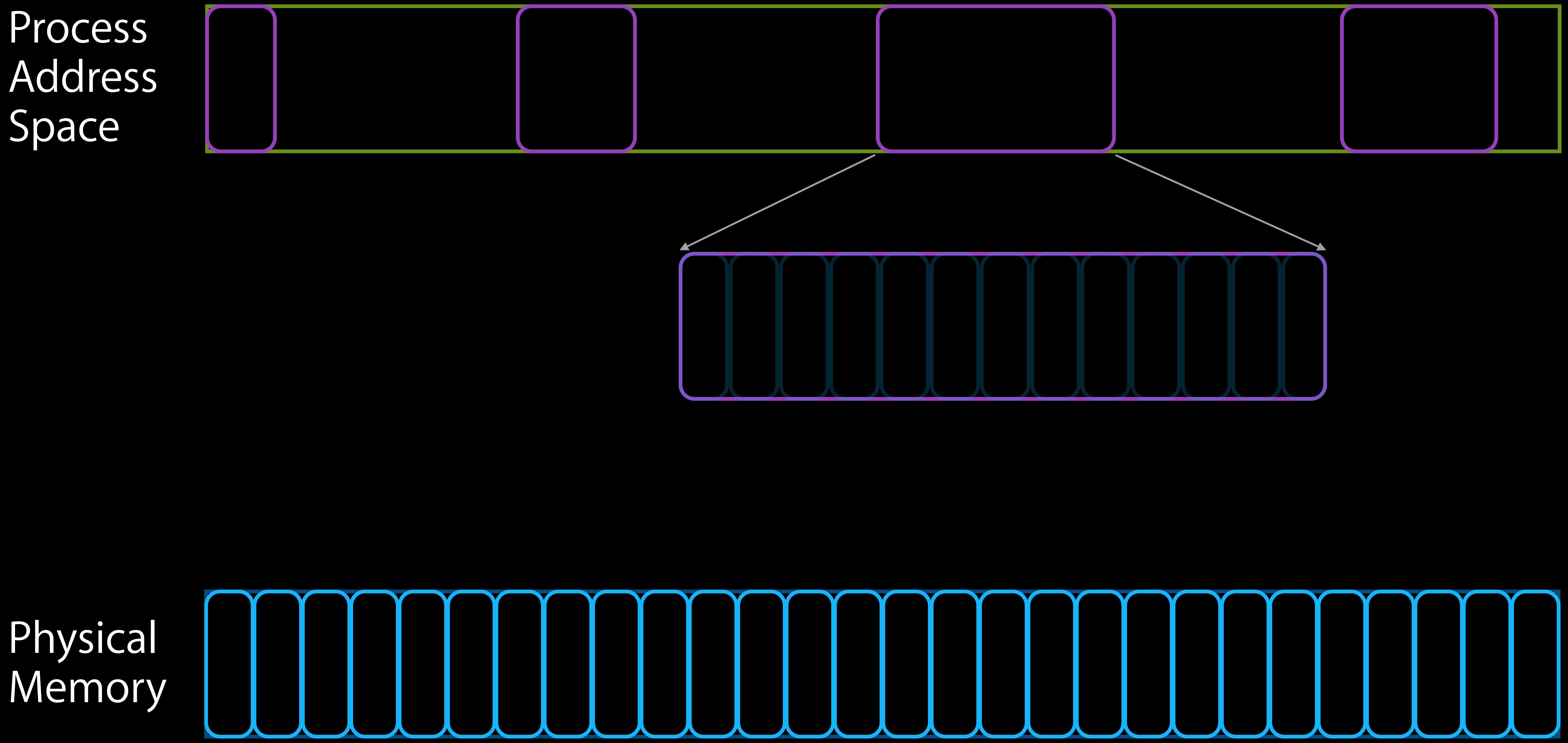
# Memory Regions



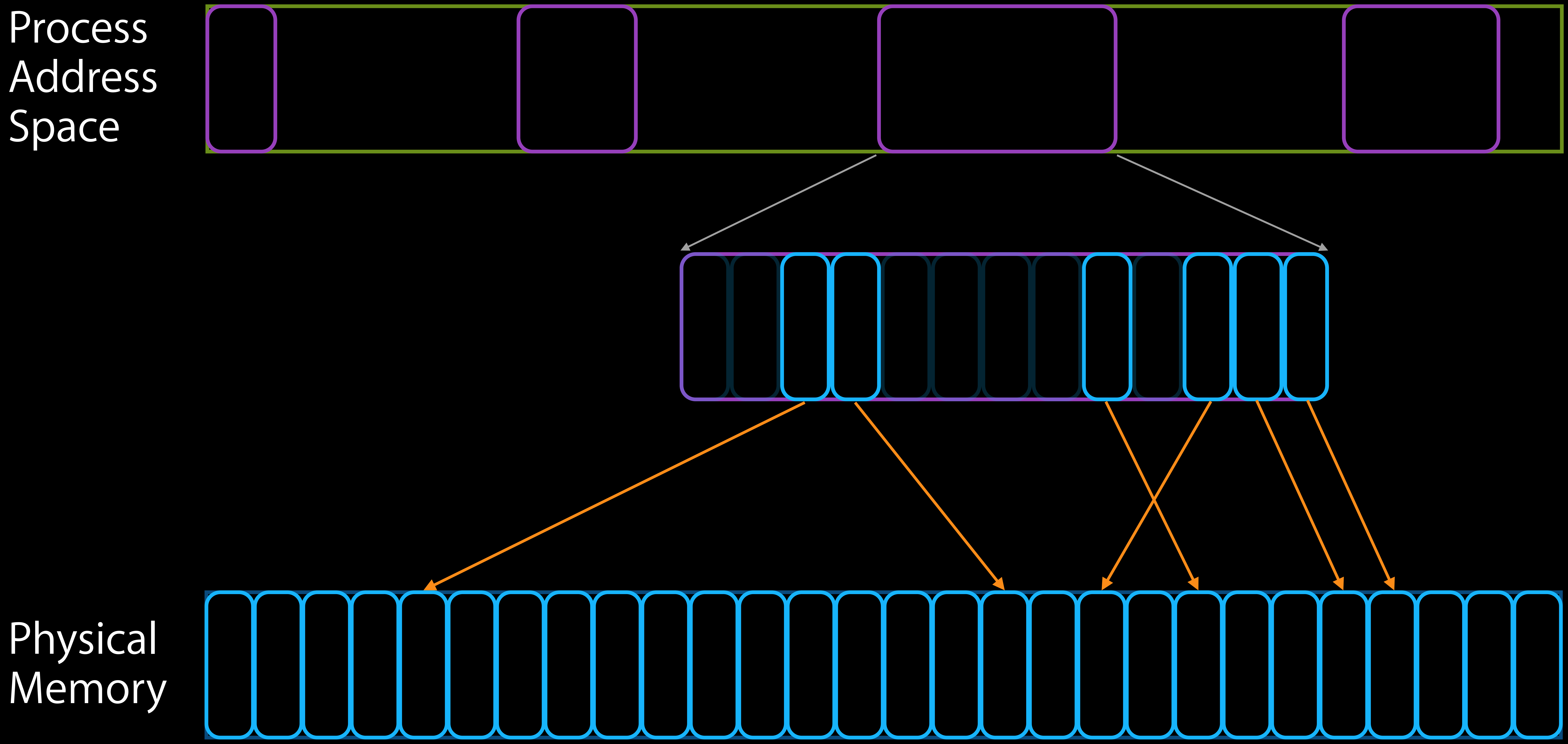
# Memory Regions



# Memory Regions

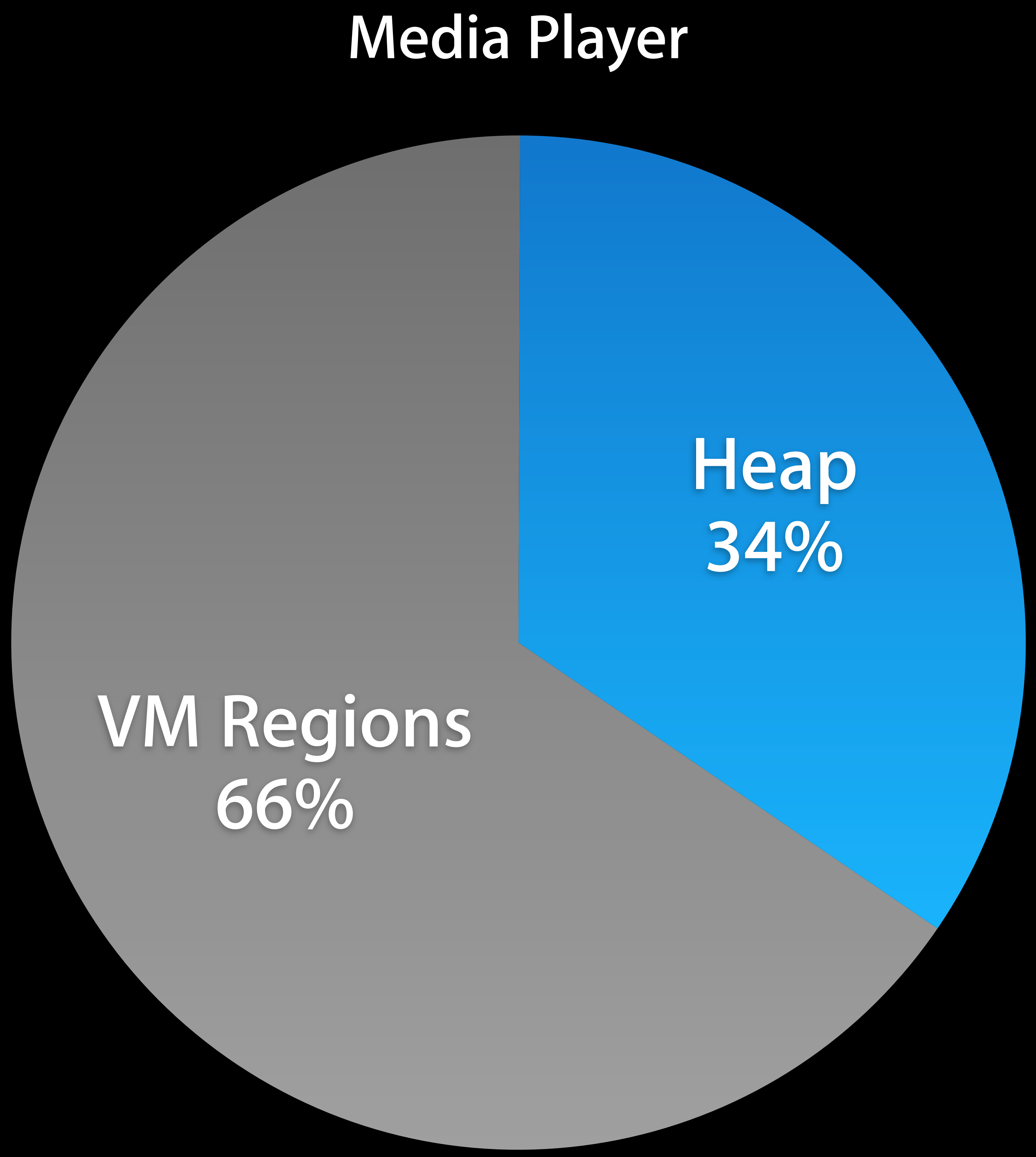


# Memory Regions

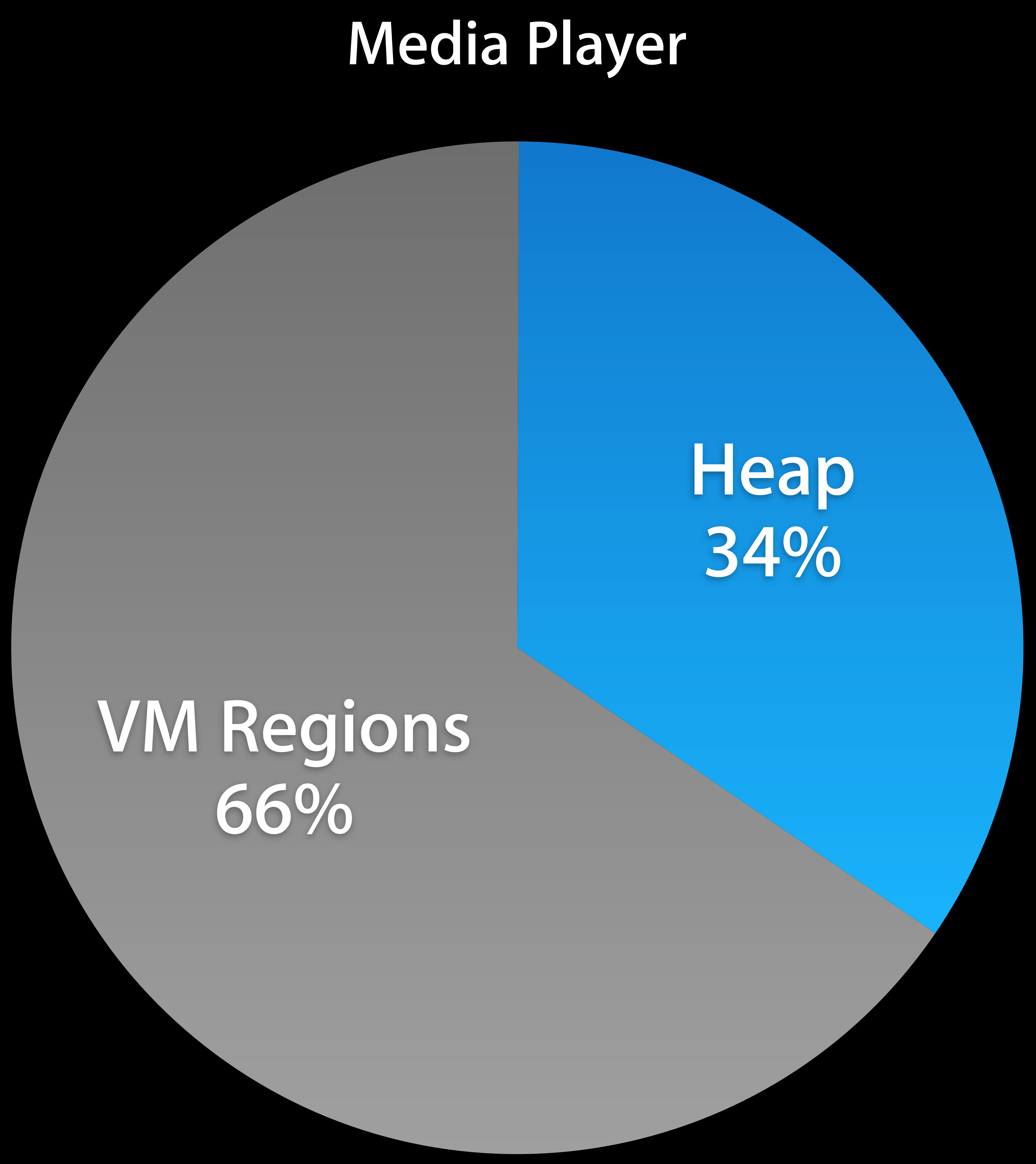


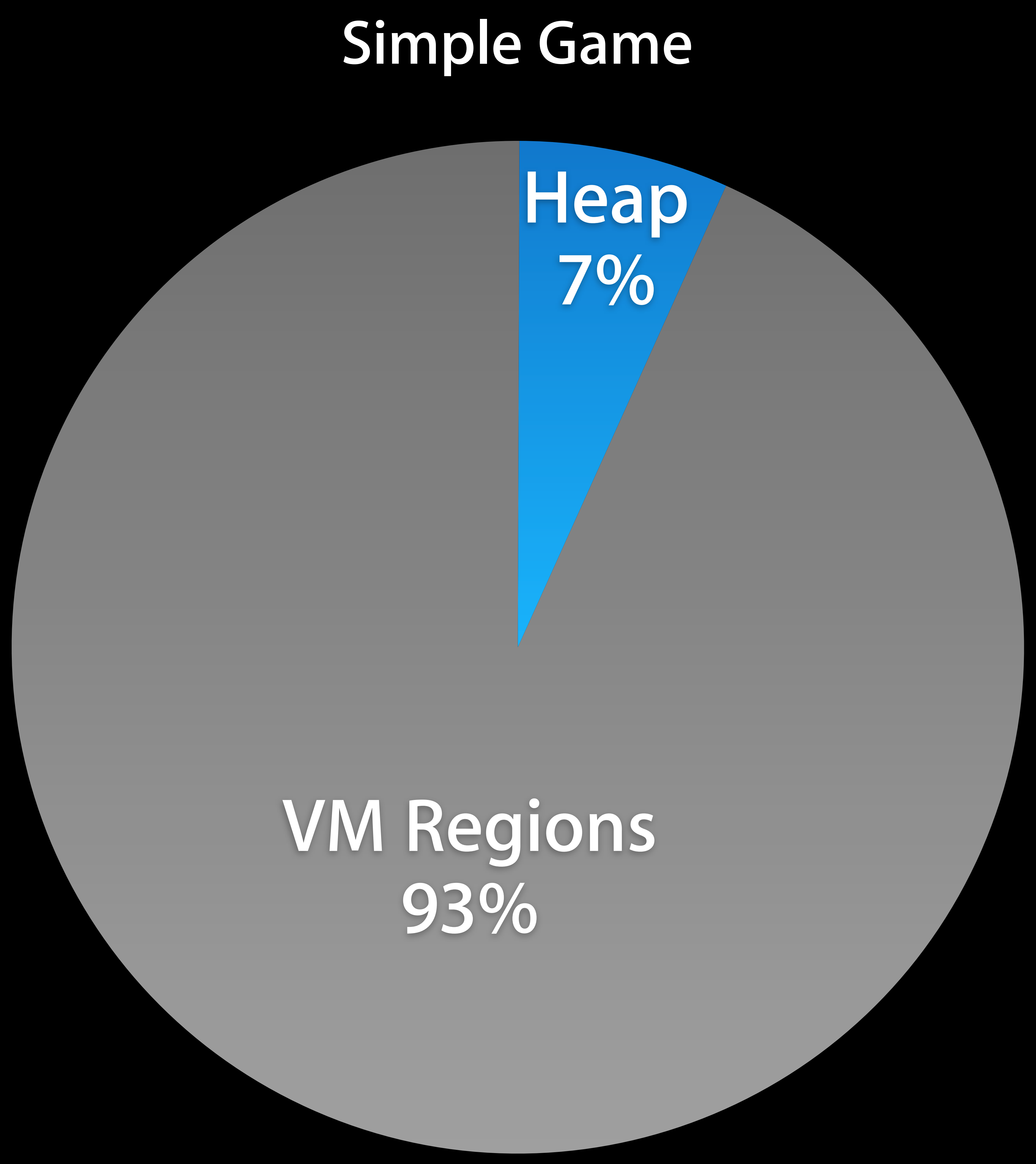
# Impact of Non-Heap Memory Regions

# Impact of Non-Heap Memory Regions



# Impact of Non-Heap Memory Regions

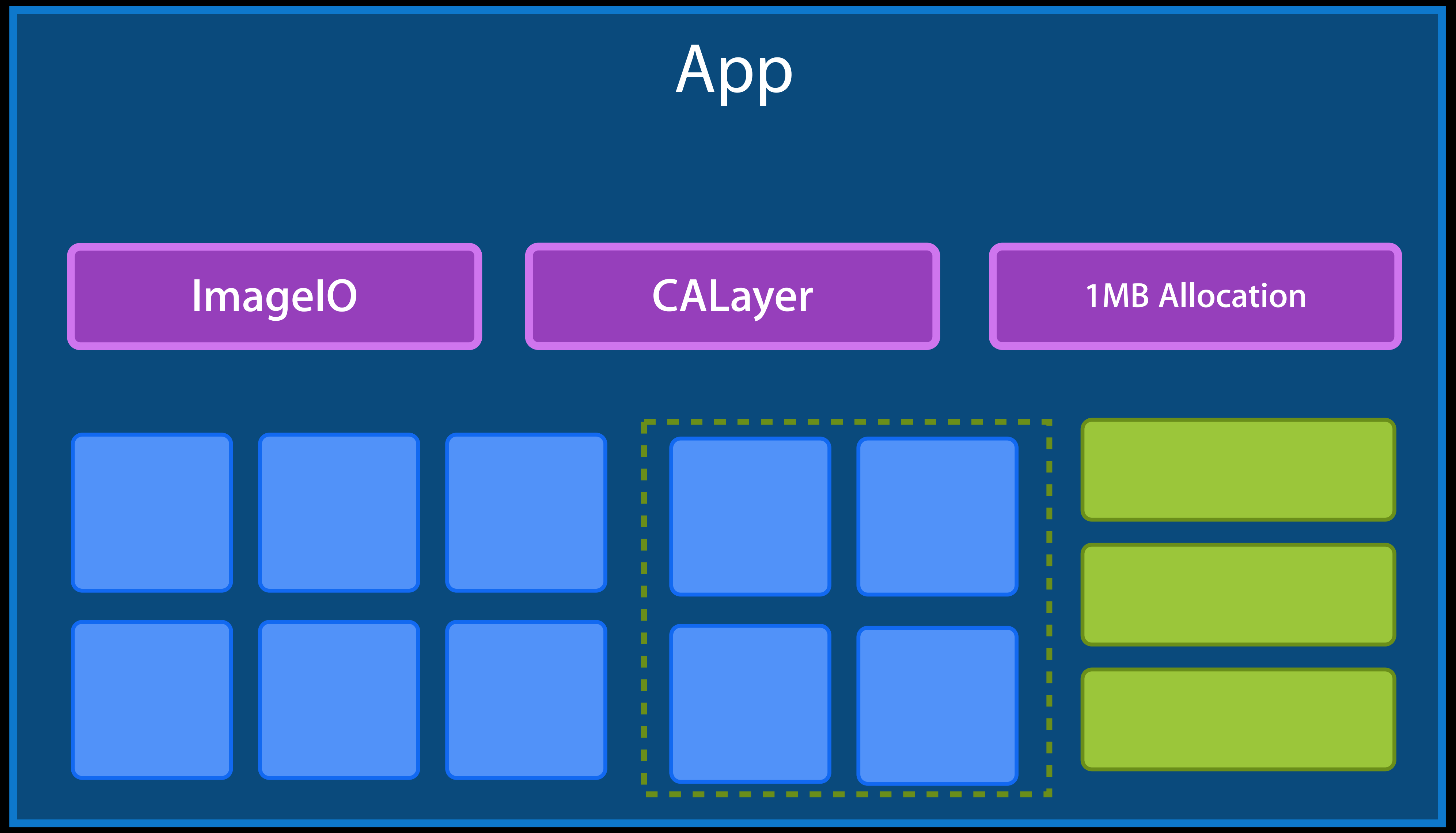




# Anonymous Memory Regions

## Common region types

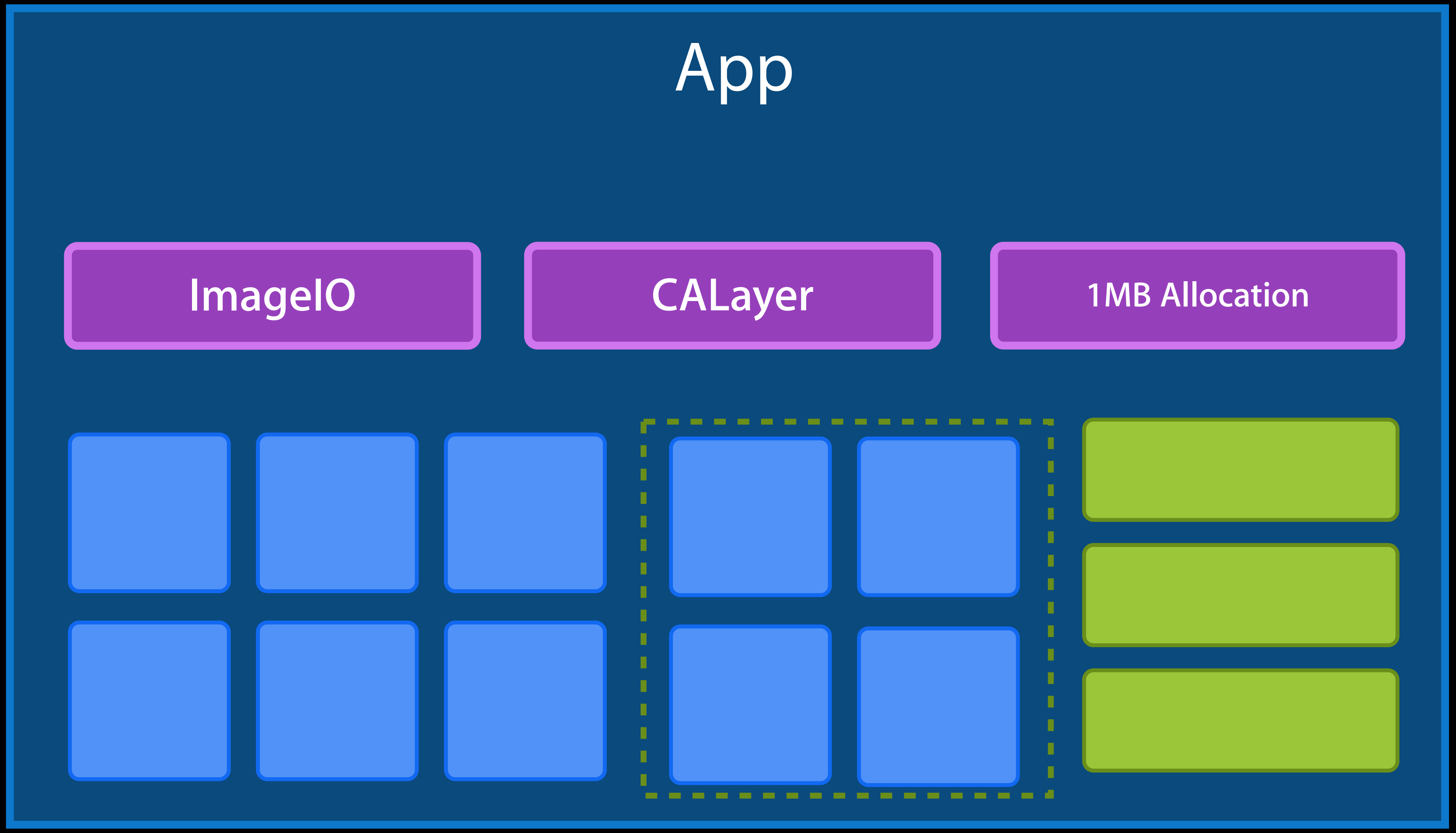
- MALLOC_SIZE—malloc blocks
- ImageIO—Decoded image data
- CALayer—Rasterized layer-backed view
  - Named for delegate



# Anonymous Memory Regions

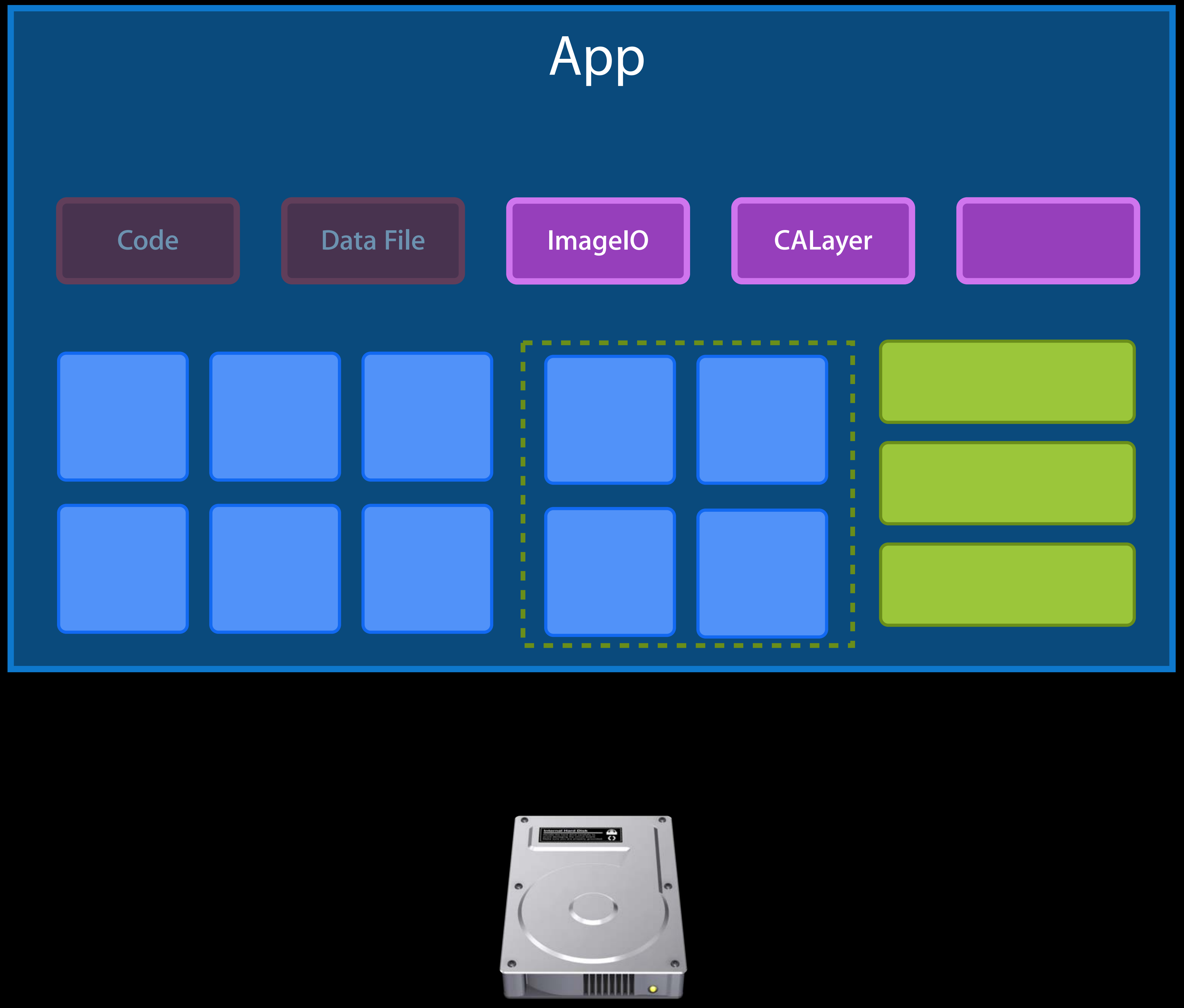
## Common region types

- MALLOC_SIZE—malloc blocks
- ImageIO—Decoded image data
- CALayer—Rasterized layer-backed view
  - Named for delegate



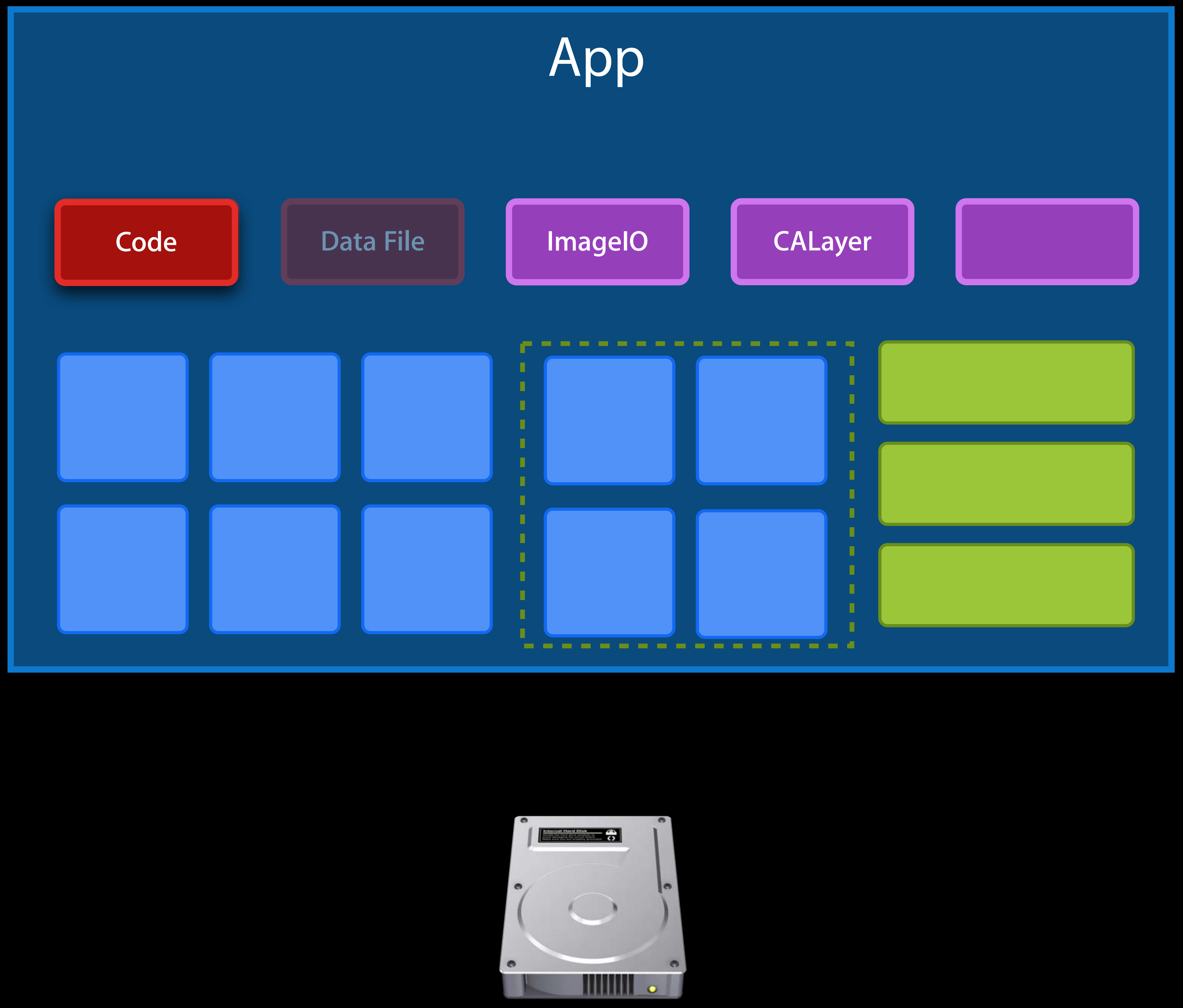
# File Backed Memory

- Regions may be backed by a file
- Data read when first accessed
- Entire region may not be resident



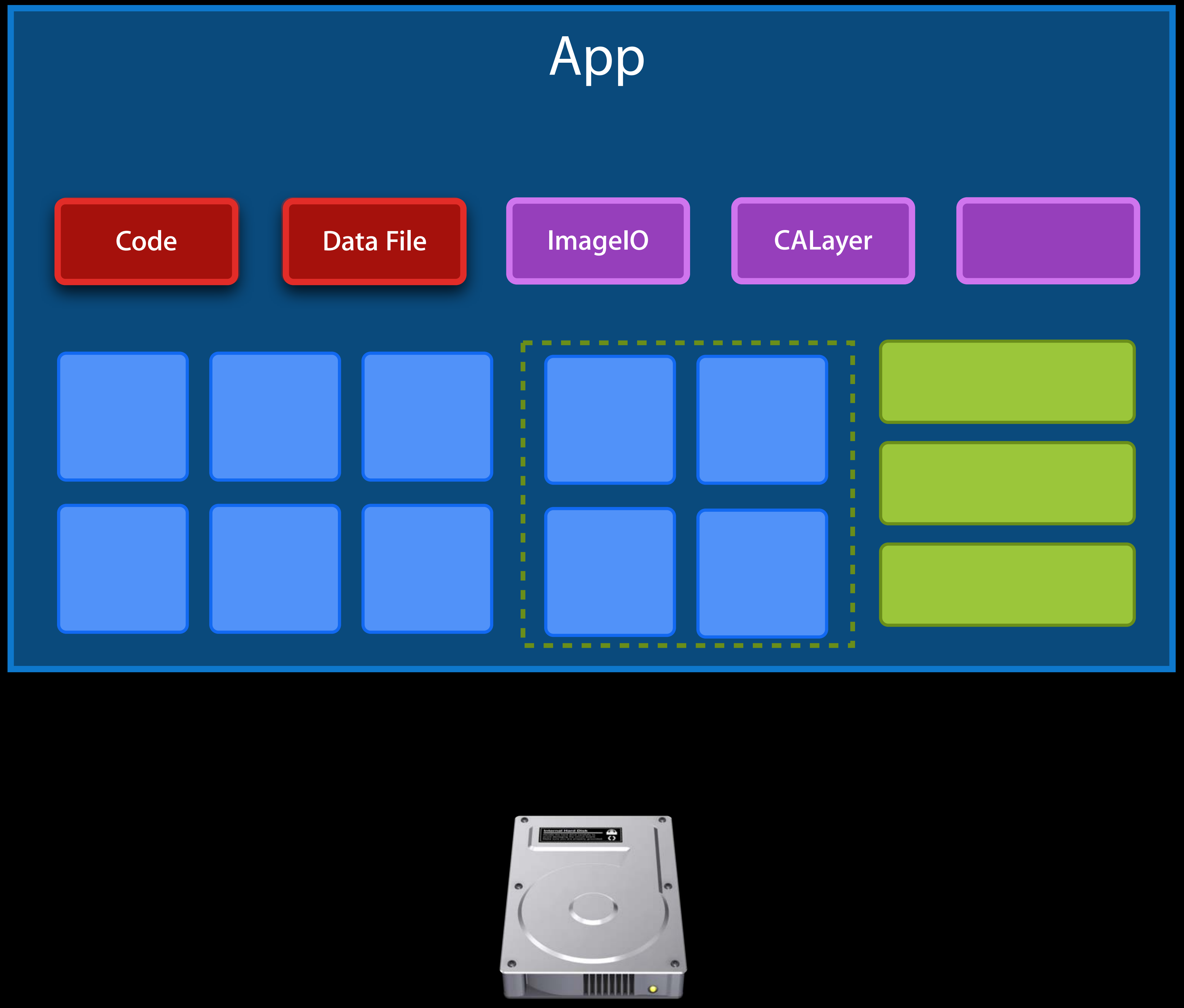
# File Backed Memory

- Regions may be backed by a file
- Data read when first accessed
- Entire region may not be resident

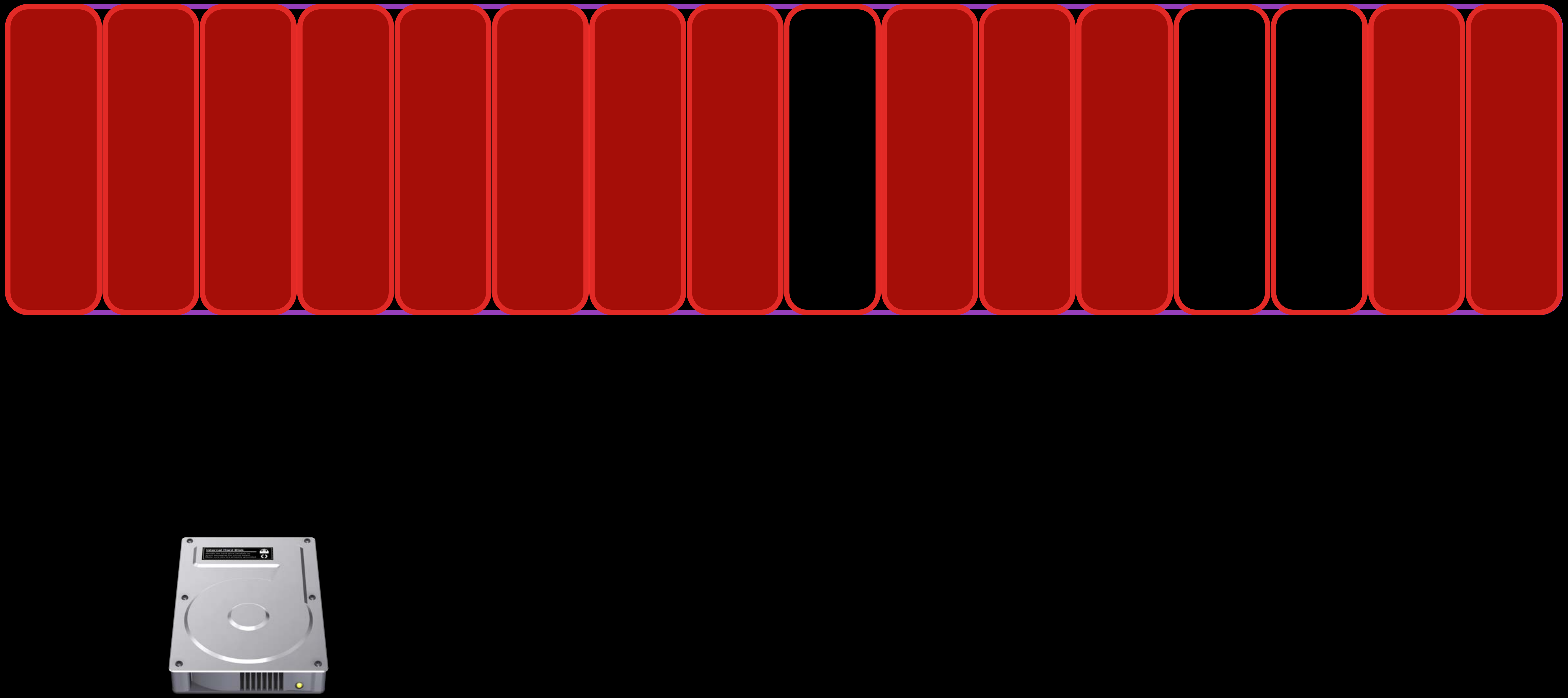


# File Backed Memory

- Regions may be backed by a file
- Data read when first accessed
- Entire region may not be resident



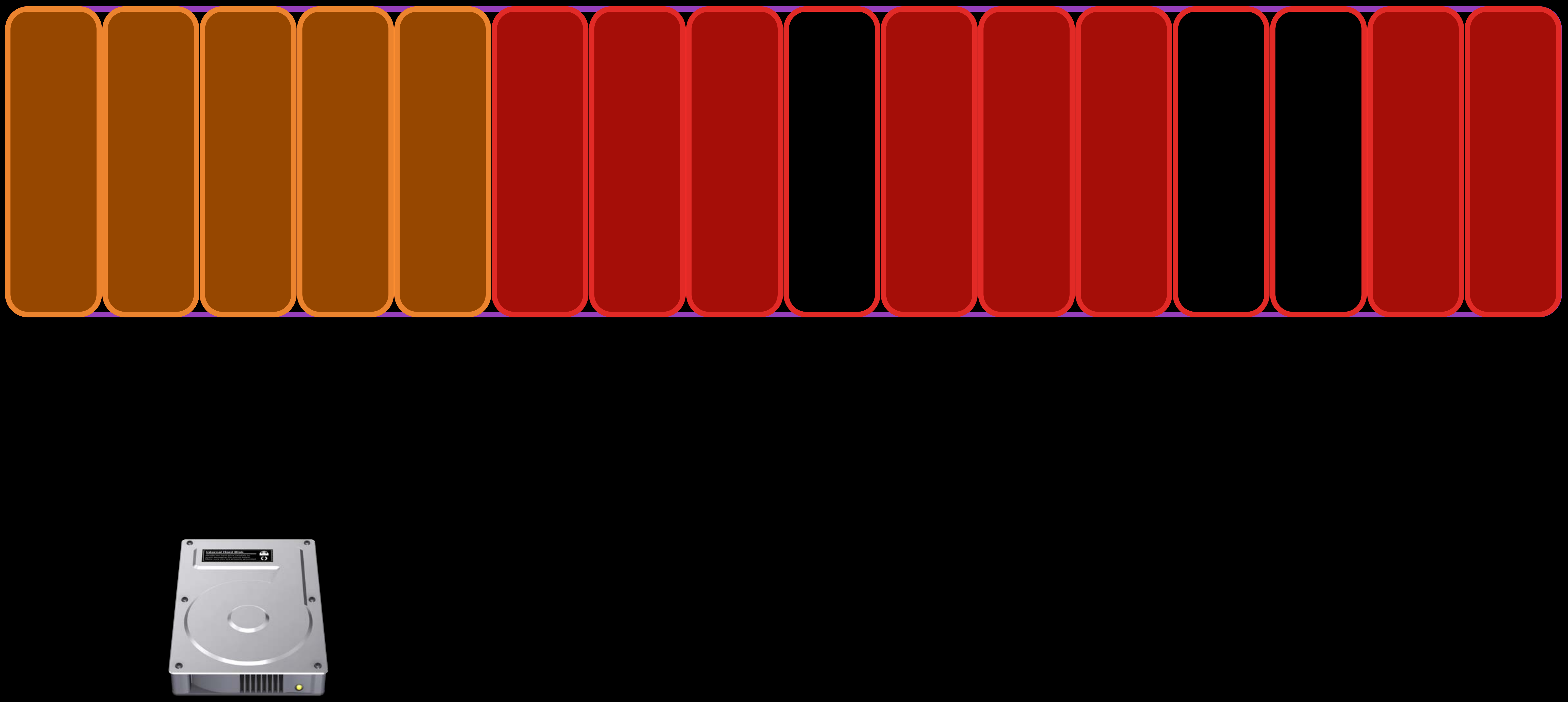
# Dirtying Memory



Writable
Shared
File-Backed
Region

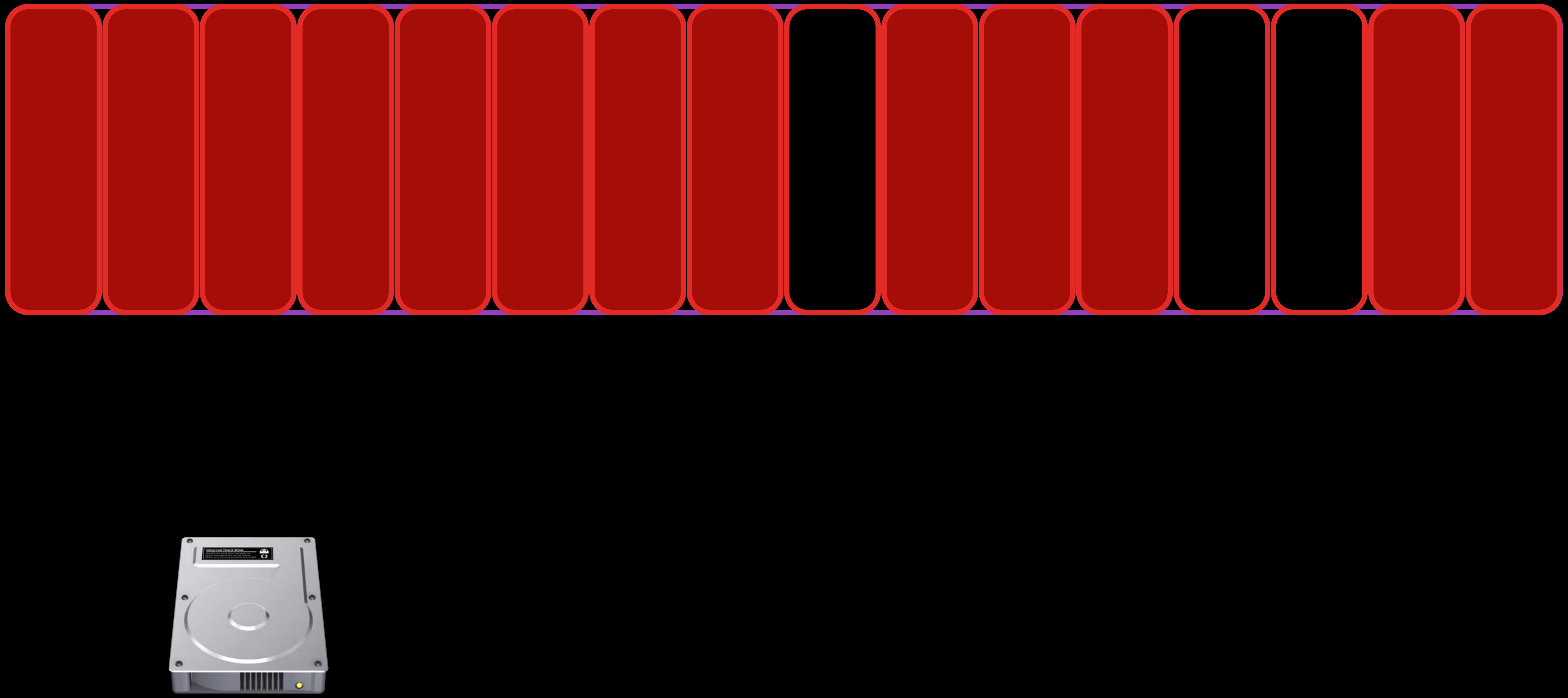
# Dirtying Memory

Writable
Shared
File-Backed
Region



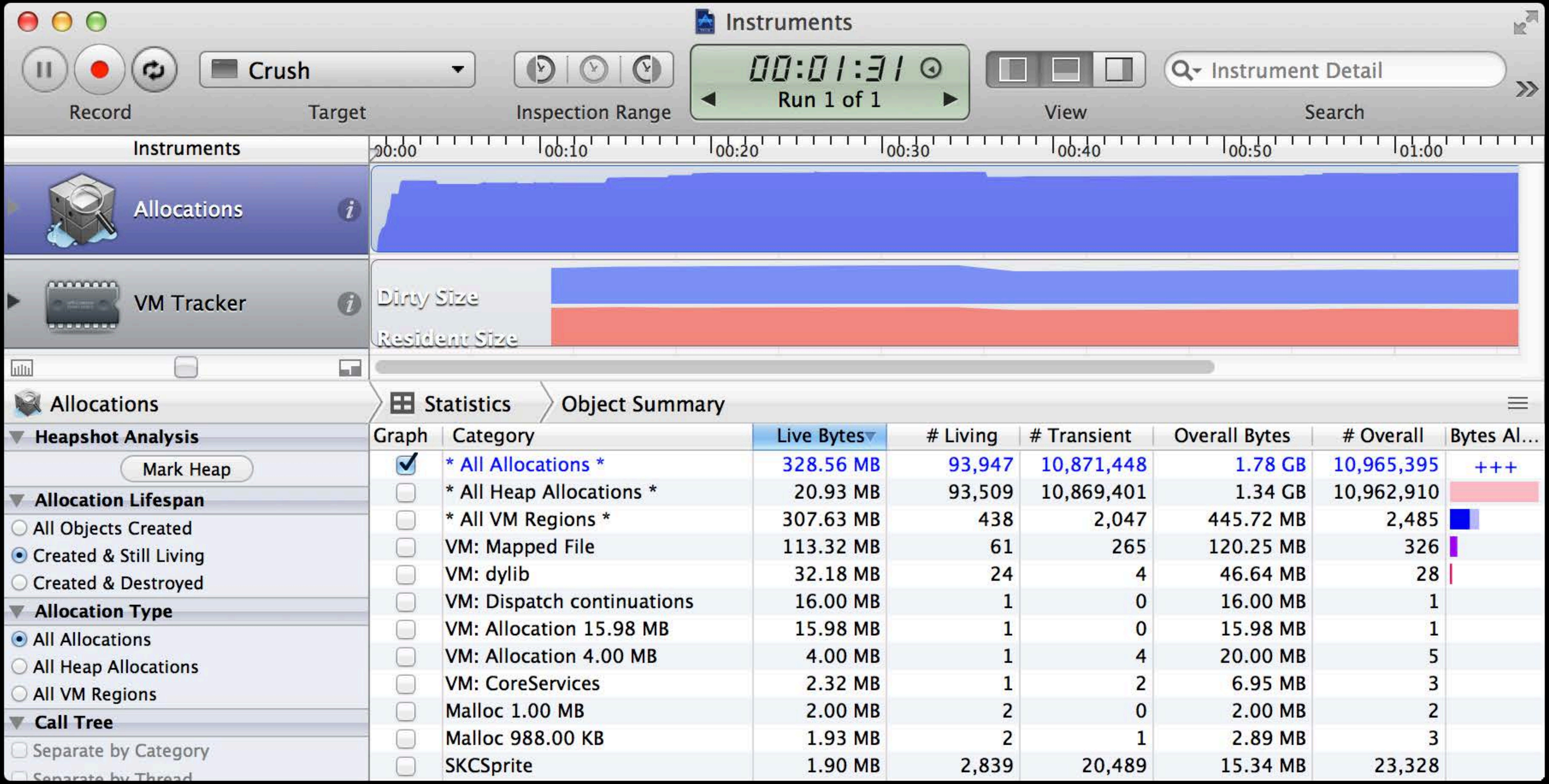
# Dirtying Memory

Writable
Shared
File-Backed
Region



# Your Memory Regions

# Your Memory Regions



**Heapshot Analysis**

Mark Heap

**Allocation Lifespan**

○ All Objects Created
● Created & Still Living
○ Created & Destroyed

**Allocation Type**

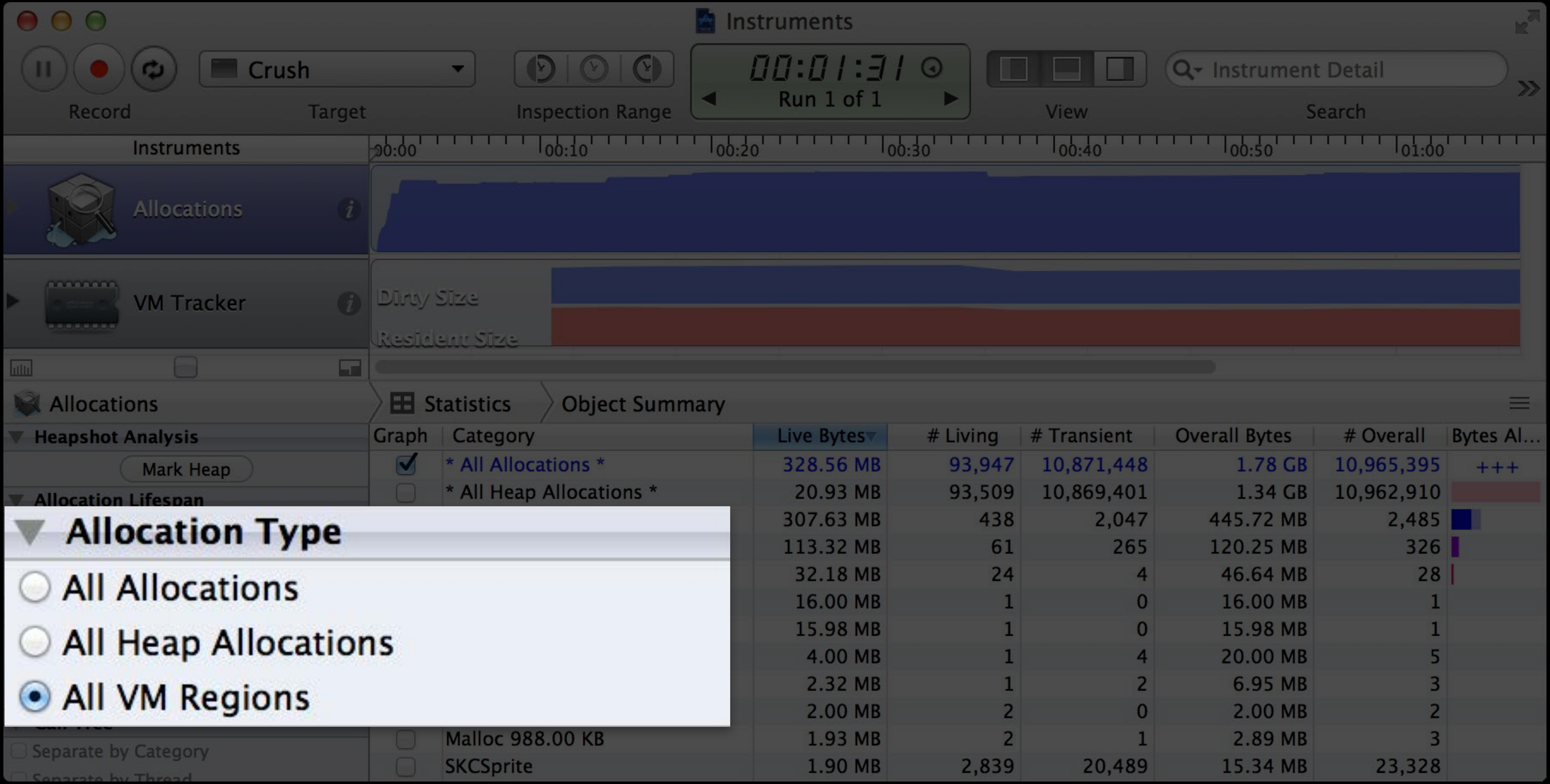
● All Allocations
○ All Heap Allocations
○ All VM Regions

**Call Tree**

☐ Separate by Category
☐ Separate by Thread

| Graph | Category | Live Bytes | # Living | # Transient | Overall Bytes | # Overall | Bytes Al... |
|---|---|---|---|---|---|---|---|
| ☑ | * All Allocations * | 328.56 MB | 93,947 | 10,871,448 | 1.78 GB | 10,965,395 | +++ |
| ☐ | * All Heap Allocations * | 20.93 MB | 93,509 | 10,869,401 | 1.34 GB | 10,962,910 | |
| ☐ | * All VM Regions * | 307.63 MB | 438 | 2,047 | 445.72 MB | 2,485 | |
| ☐ | VM: Mapped File | 113.32 MB | 61 | 265 | 120.25 MB | 326 | |
| ☐ | VM: dylib | 32.18 MB | 24 | 4 | 46.64 MB | 28 | |
| ☐ | VM: Dispatch continuations | 16.00 MB | 1 | 0 | 16.00 MB | 1 | |
| ☐ | VM: Allocation 15.98 MB | 15.98 MB | 1 | 0 | 15.98 MB | 1 | |
| ☐ | VM: Allocation 4.00 MB | 4.00 MB | 1 | 4 | 20.00 MB | 5 | |
| ☐ | VM: CoreServices | 2.32 MB | 1 | 2 | 6.95 MB | 3 | |
| ☐ | Malloc 1.00 MB | 2.00 MB | 2 | 0 | 2.00 MB | 2 | |
| ☐ | Malloc 988.00 KB | 1.93 MB | 2 | 1 | 2.89 MB | 3 | |
| ☐ | SKCSprite | 1.90 MB | 2,839 | 20,489 | 15.34 MB | 23,328 | |

# Your Memory Regions



**Allocation Type**

○ All Allocations

○ All Heap Allocations

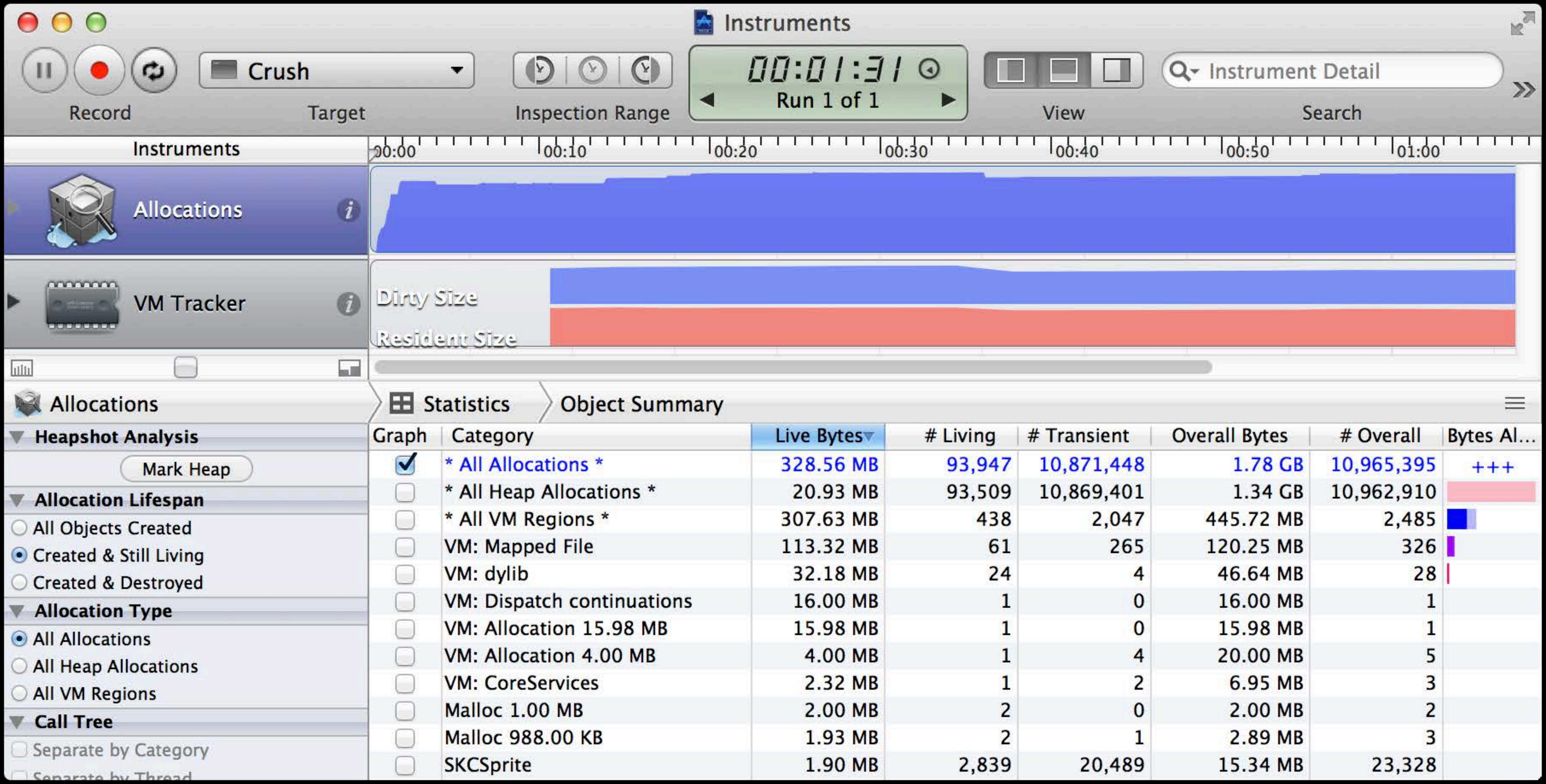
● All VM Regions

□ Separate by Category

□ Separate by Thread

| Graph | Category | Live Bytes | # Living | # Transient | Overall Bytes | # Overall | Bytes Al... |
|---|---|---|---|---|---|---|---|
| ☑ | * All Allocations * | 328.56 MB | 93,947 | 10,871,448 | 1.78 GB | 10,965,395 | +++ |
| □ | * All Heap Allocations * | 20.93 MB | 93,509 | 10,869,401 | 1.34 GB | 10,962,910 | |
| | | 307.63 MB | 438 | 2,047 | 445.72 MB | 2,485 | |
| | | 113.32 MB | 61 | 265 | 120.25 MB | 326 | |
| | | 32.18 MB | 24 | 4 | 46.64 MB | 28 | |
| | | 16.00 MB | 1 | 0 | 16.00 MB | 1 | |
| | | 15.98 MB | 1 | 0 | 15.98 MB | 1 | |
| | | 4.00 MB | 1 | 4 | 20.00 MB | 5 | |
| | | 2.32 MB | 1 | 2 | 6.95 MB | 3 | |
| | | 2.00 MB | 2 | 0 | 2.00 MB | 2 | |
| □ | Malloc 988.00 KB | 1.93 MB | 2 | 1 | 2.89 MB | 3 | |
| □ | SKCSprite | 1.90 MB | 2,839 | 20,489 | 15.34 MB | 23,328 | |

# Your Memory Regions



Allocations

**Heapshot Analysis**

Mark Heap

**Allocation Lifespan**

- ○ All Objects Created
- ● Created & Still Living
- ○ Created & Destroyed

**Allocation Type**

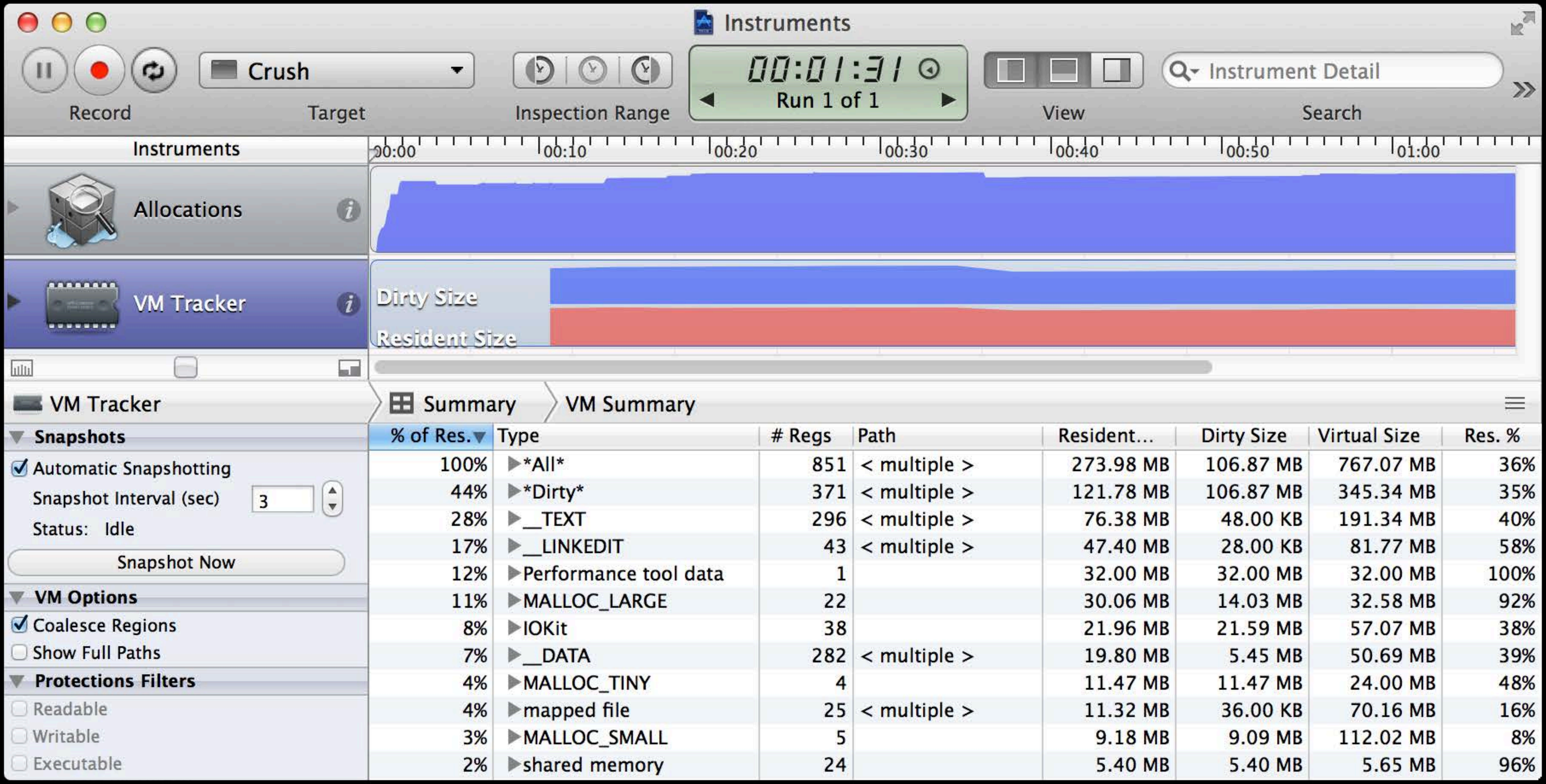
- ● All Allocations
- ○ All Heap Allocations
- ○ All VM Regions

**Call Tree**

- □ Separate by Category
- □ Separate by Thread

| Graph | Category | Live Bytes | # Living | # Transient | Overall Bytes | # Overall | Bytes Al... |
|---|---|---|---|---|---|---|---|
| ☑ | * All Allocations * | 328.56 MB | 93,947 | 10,871,448 | 1.78 GB | 10,965,395 | +++ |
| □ | * All Heap Allocations * | 20.93 MB | 93,509 | 10,869,401 | 1.34 GB | 10,962,910 | |
| □ | * All VM Regions * | 307.63 MB | 438 | 2,047 | 445.72 MB | 2,485 | |
| □ | VM: Mapped File | 113.32 MB | 61 | 265 | 120.25 MB | 326 | |
| □ | VM: dylib | 32.18 MB | 24 | 4 | 46.64 MB | 28 | |
| □ | VM: Dispatch continuations | 16.00 MB | 1 | 0 | 16.00 MB | 1 | |
| □ | VM: Allocation 15.98 MB | 15.98 MB | 1 | 0 | 15.98 MB | 1 | |
| □ | VM: Allocation 4.00 MB | 4.00 MB | 1 | 4 | 20.00 MB | 5 | |
| □ | VM: CoreServices | 2.32 MB | 1 | 2 | 6.95 MB | 3 | |
| □ | Malloc 1.00 MB | 2.00 MB | 2 | 0 | 2.00 MB | 2 | |
| □ | Malloc 988.00 KB | 1.93 MB | 2 | 1 | 2.89 MB | 3 | |
| □ | SKCSprite | 1.90 MB | 2,839 | 20,489 | 15.34 MB | 23,328 | |

# Your Memory Regions



VM Tracker

**Snapshots**

☑ Automatic Snapshotting

Snapshot Interval (sec) 3

Status: Idle

Snapshot Now

**VM Options**

☑ Coalesce Regions

□ Show Full Paths

**Protections Filters**

□ Readable

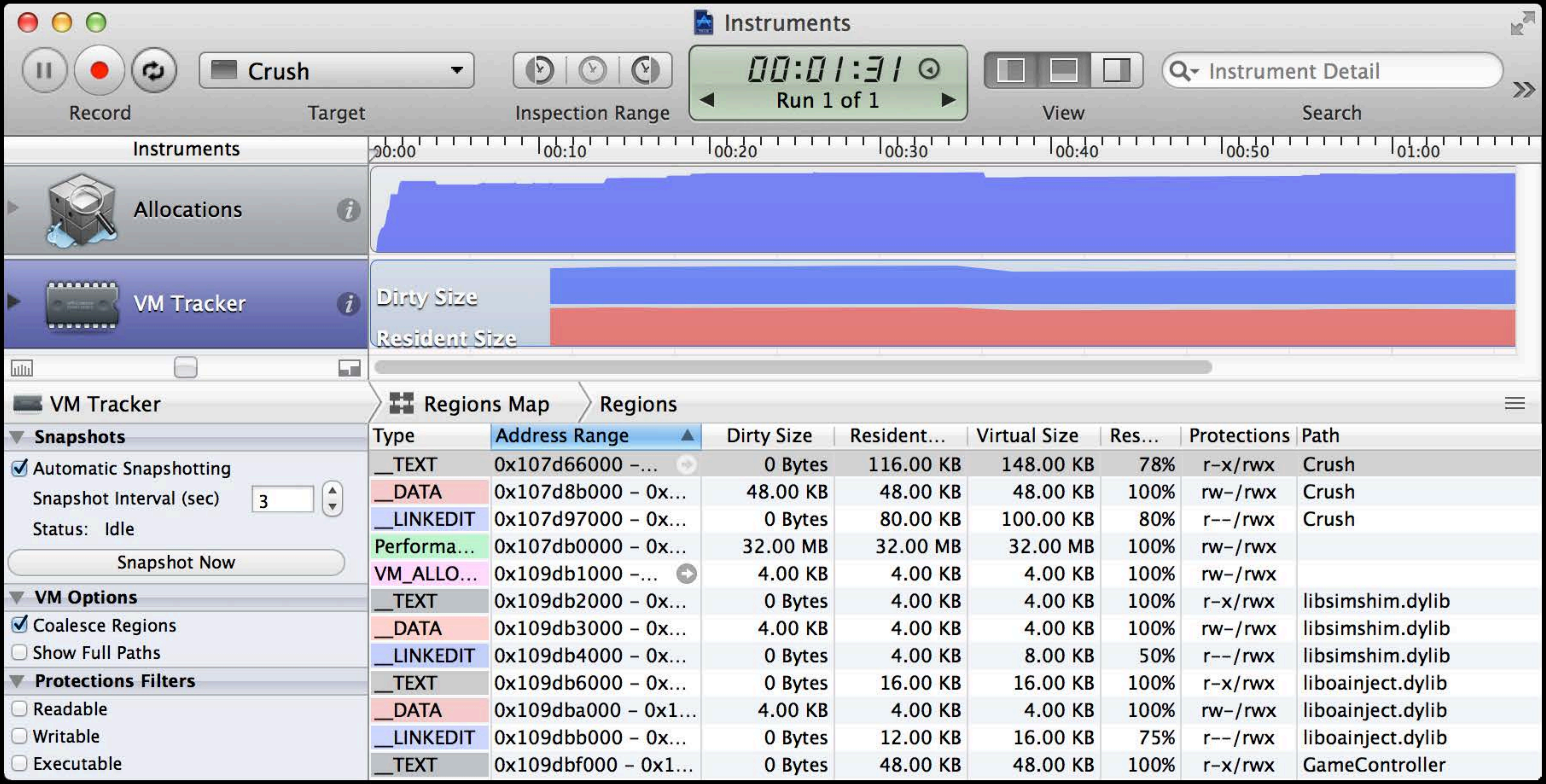
□ Writable

□ Executable

Summary > VM Summary

| % of Res. | Type | # Regs | Path | Resident... | Dirty Size | Virtual Size | Res. % |
|---|---|---|---|---|---|---|---|
| 100% | ▶*All* | 851 | < multiple > | 273.98 MB | 106.87 MB | 767.07 MB | 36% |
| 44% | ▶*Dirty* | 371 | < multiple > | 121.78 MB | 106.87 MB | 345.34 MB | 35% |
| 28% | ▶__TEXT | 296 | < multiple > | 76.38 MB | 48.00 KB | 191.34 MB | 40% |
| 17% | ▶__LINKEDIT | 43 | < multiple > | 47.40 MB | 28.00 KB | 81.77 MB | 58% |
| 12% | ▶Performance tool data | 1 | | 32.00 MB | 32.00 MB | 32.00 MB | 100% |
| 11% | ▶MALLOC_LARGE | 22 | | 30.06 MB | 14.03 MB | 32.58 MB | 92% |
| 8% | ▶IOKit | 38 | | 21.96 MB | 21.59 MB | 57.07 MB | 38% |
| 7% | ▶__DATA | 282 | < multiple > | 19.80 MB | 5.45 MB | 50.69 MB | 39% |
| 4% | ▶MALLOC_TINY | 4 | | 11.47 MB | 11.47 MB | 24.00 MB | 48% |
| 4% | ▶mapped file | 25 | < multiple > | 11.32 MB | 36.00 KB | 70.16 MB | 16% |
| 3% | ▶MALLOC_SMALL | 5 | | 9.18 MB | 9.09 MB | 112.02 MB | 8% |
| 2% | ▶shared memory | 24 | | 5.40 MB | 5.40 MB | 5.65 MB | 96% |

# Your Memory Regions



**Snapshots**

☑ Automatic Snapshotting

Snapshot Interval (sec) 3

Status: Idle

Snapshot Now

**VM Options**

☑ Coalesce Regions

□ Show Full Paths

**Protections Filters**

□ Readable

□ Writable

□ Executable

| Type | Address Range | Dirty Size | Resident... | Virtual Size | Res... | Protections | Path |
|---|---|---|---|---|---|---|---|
| __TEXT | 0x107d66000 –... | 0 Bytes | 116.00 KB | 148.00 KB | 78% | r-x/rwx | Crush |
| __DATA | 0x107d8b000 – 0x... | 48.00 KB | 48.00 KB | 48.00 KB | 100% | rw-/rwx | Crush |
| __LINKEDIT | 0x107d97000 – 0x... | 0 Bytes | 80.00 KB | 100.00 KB | 80% | r--/rwx | Crush |
| Performa... | 0x107db0000 – 0x... | 32.00 MB | 32.00 MB | 32.00 MB | 100% | rw-/rwx | |
| VM_ALLO... | 0x109db1000 –... | 4.00 KB | 4.00 KB | 4.00 KB | 100% | rw-/rwx | |
| __TEXT | 0x109db2000 – 0x... | 0 Bytes | 4.00 KB | 4.00 KB | 100% | r-x/rwx | libsimshim.dylib |
| __DATA | 0x109db3000 – 0x... | 4.00 KB | 4.00 KB | 4.00 KB | 100% | rw-/rwx | libsimshim.dylib |
| __LINKEDIT | 0x109db4000 – 0x... | 0 Bytes | 4.00 KB | 8.00 KB | 50% | r--/rwx | libsimshim.dylib |
| __TEXT | 0x109db6000 – 0x... | 0 Bytes | 16.00 KB | 16.00 KB | 100% | r-x/rwx | liboainject.dylib |
| __DATA | 0x109dba000 – 0x1... | 4.00 KB | 4.00 KB | 4.00 KB | 100% | rw-/rwx | liboainject.dylib |
| __LINKEDIT | 0x109dbb000 – 0x... | 0 Bytes | 12.00 KB | 16.00 KB | 75% | r--/rwx | liboainject.dylib |
| __TEXT | 0x109dbf000 – 0x1... | 0 Bytes | 48.00 KB | 48.00 KB | 100% | r-x/rwx | GameController |

# Measuring App Footprint

# Measuring App Footprint

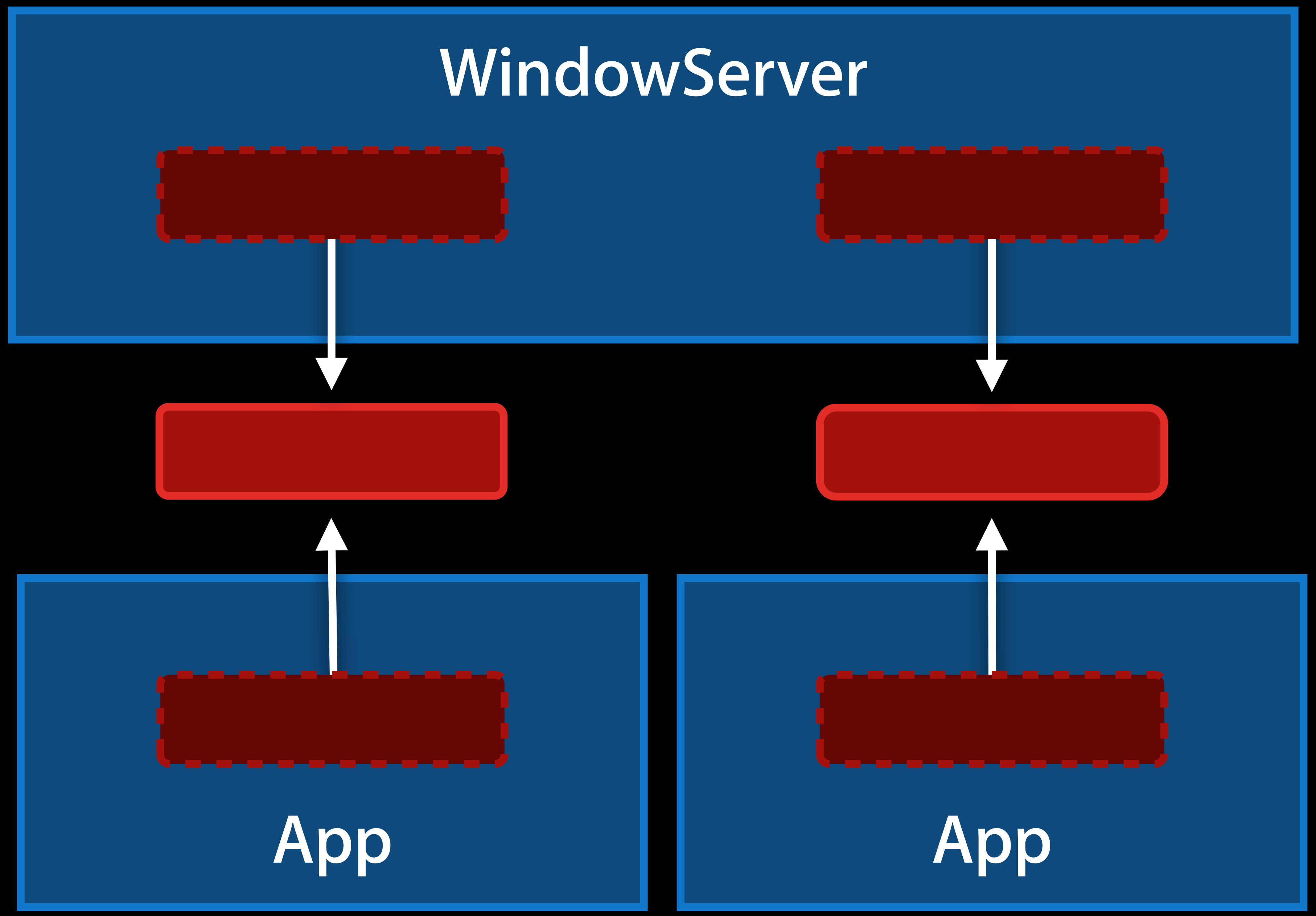
```
$ sudo footprint -proc MyLeakyApp -swapped -categories
```

# Measuring App Footprint

```
$ sudo footprint -proc MyLeakyApp -swapped -categories

================================================================================
  MyLeakyApp [8470]: 64-bit   Footprint: 12.01 MB
================================================================================

--------------------------------------------------------------------------------
                            Contributes to Footprint
--------------------------------------------------------------------------------

      5704 kB           Private Dirty                          (2116 kB swapped)
          3772 kB           Malloc Memory                      (1696 kB swapped)
              1544 kB           MALLOC_TINY                     (348 kB swapped)
```

# Shared Memory

- Memory regions may be shared
  - Used for graphics memory
  - Common in multi-process apps
- May not be visible in Allocations



# Measuring Multi-Process Footprint

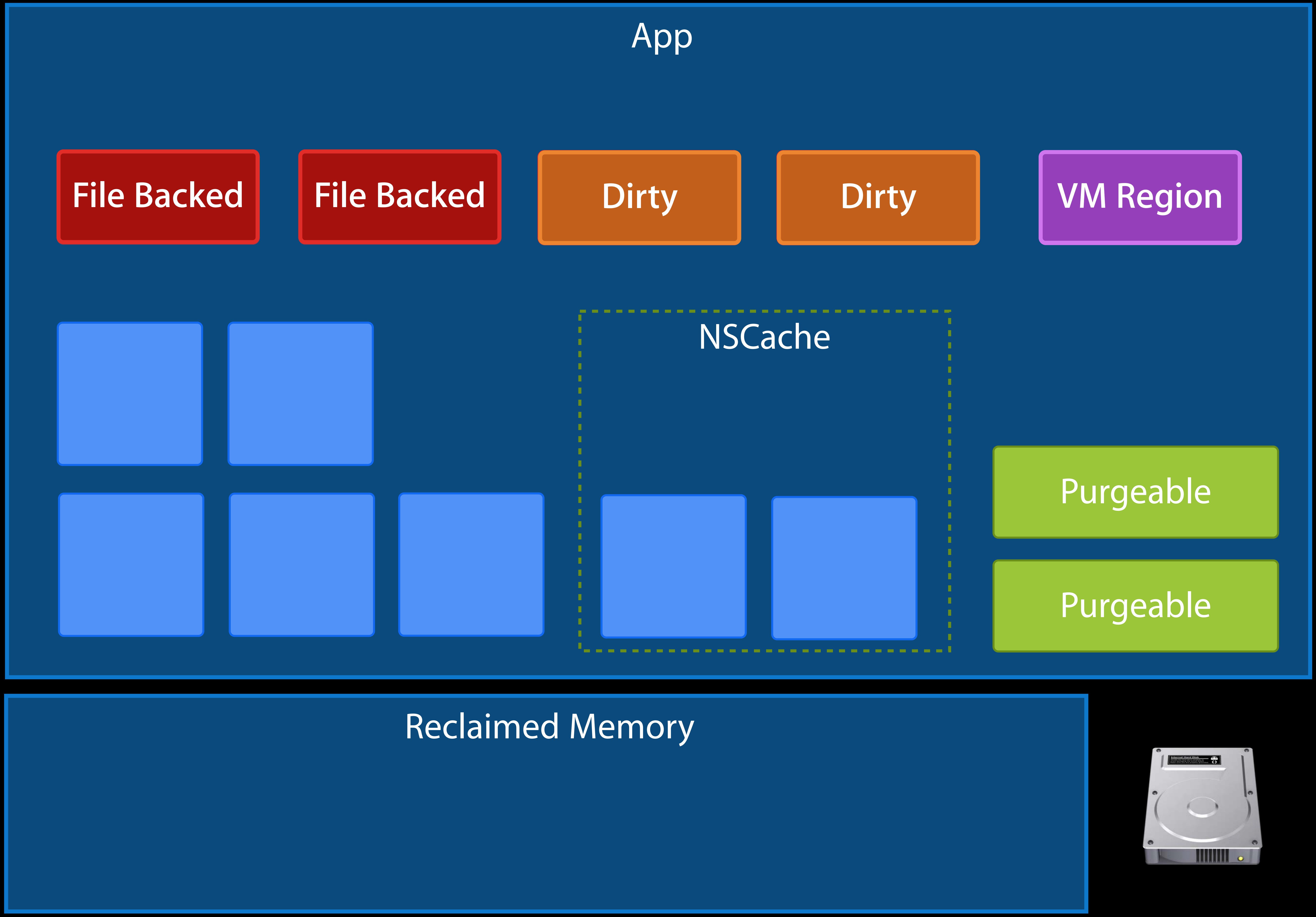
# Measuring Multi-Process Footprint

```
$ sudo footprint -proc <App> -proc WindowServer
```

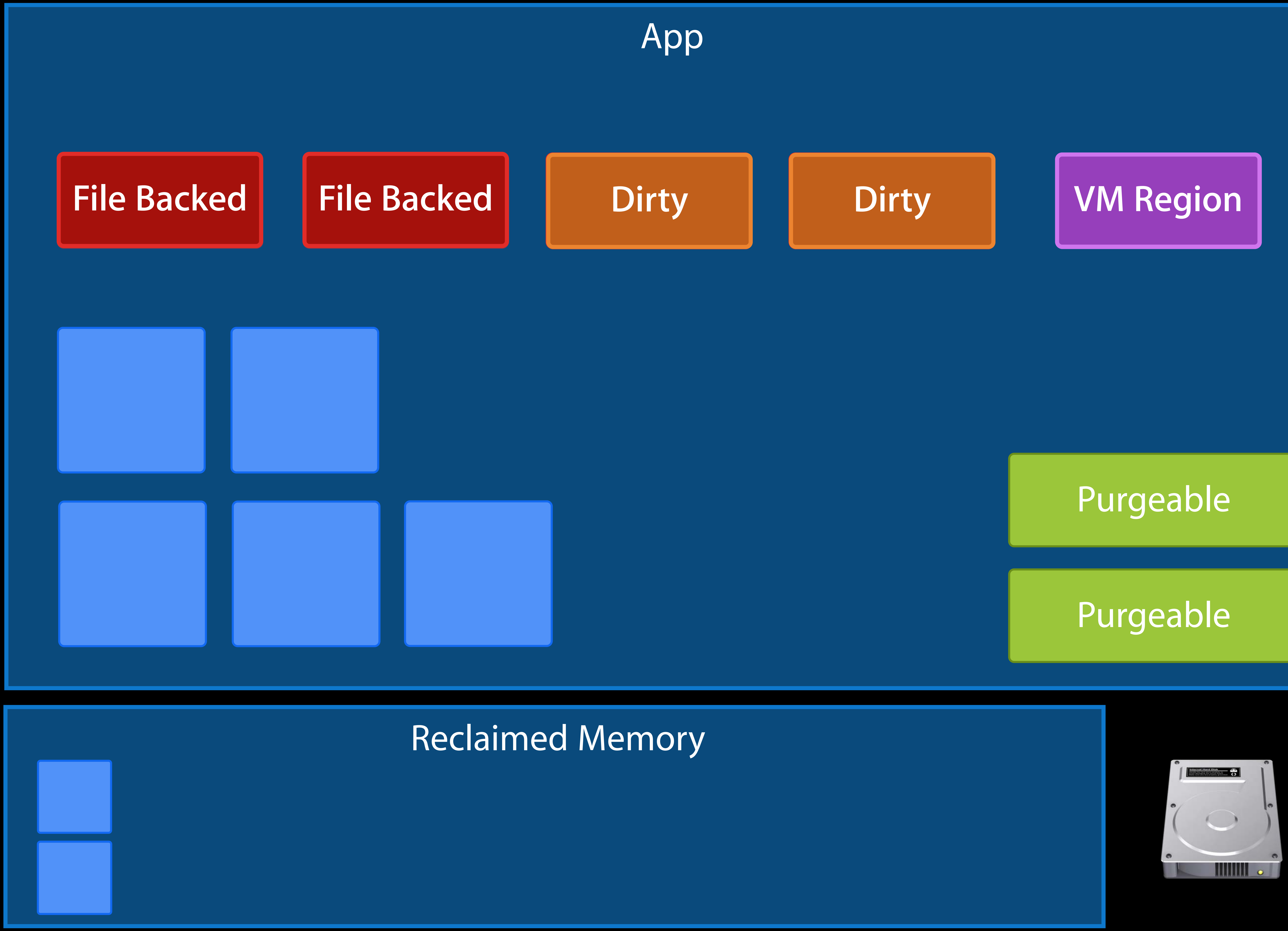
# Measuring Multi-Process Footprint

```
$ sudo footprint -proc <App> -proc WindowServer

       28.46 MB           Shared Dirty
        8032 kB           With WindowServer [96]
          4192 kB           Other
          3840 kB           CoreGraphics-related Memory
       20.62 MB           With Others
         20.12 MB           Other
           324 kB           Malloc Memory
           104 kB           Application-specific Memory

...

  622.36 MB           Total footprint
```

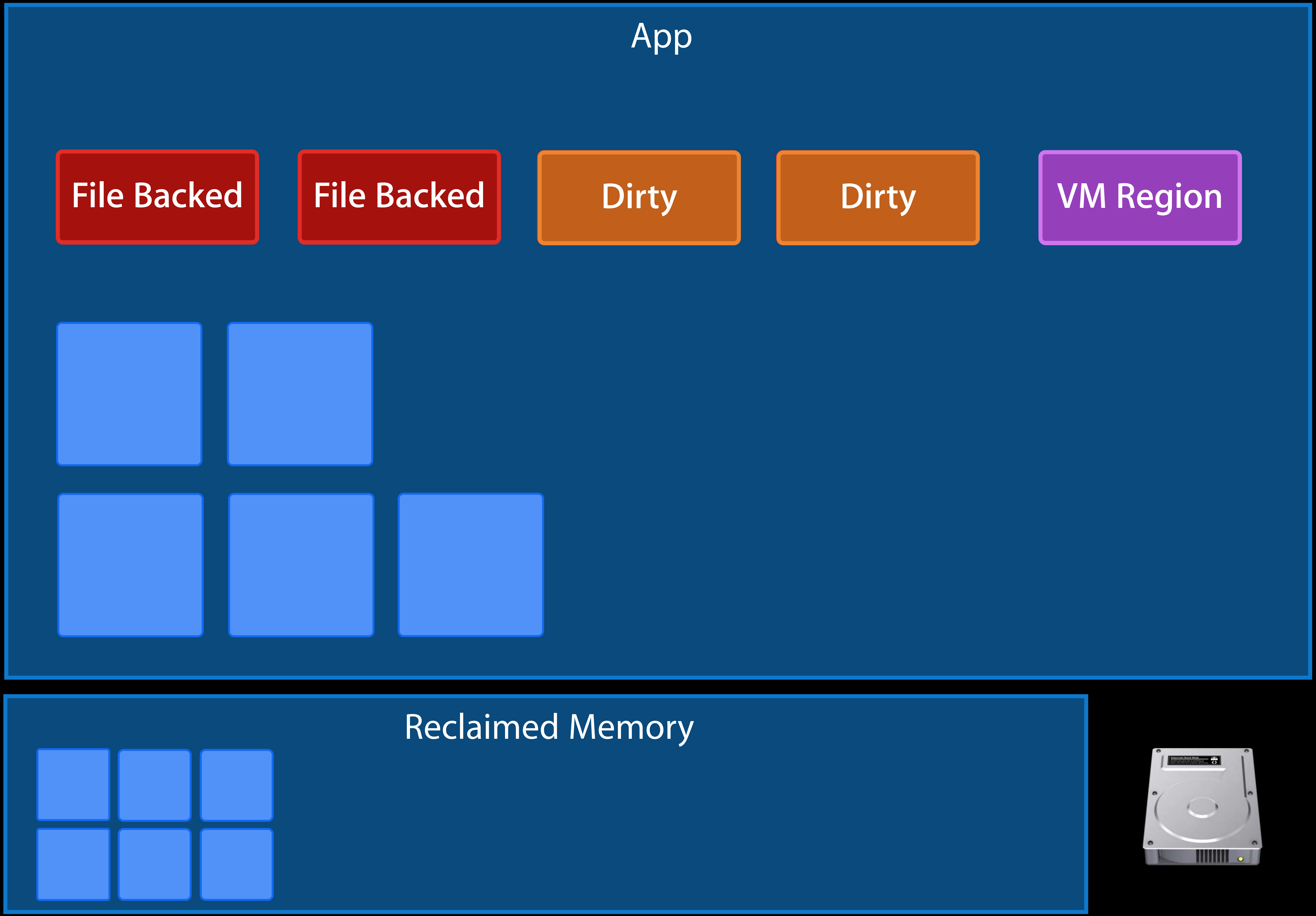
# Satisfying Demand for New Pages



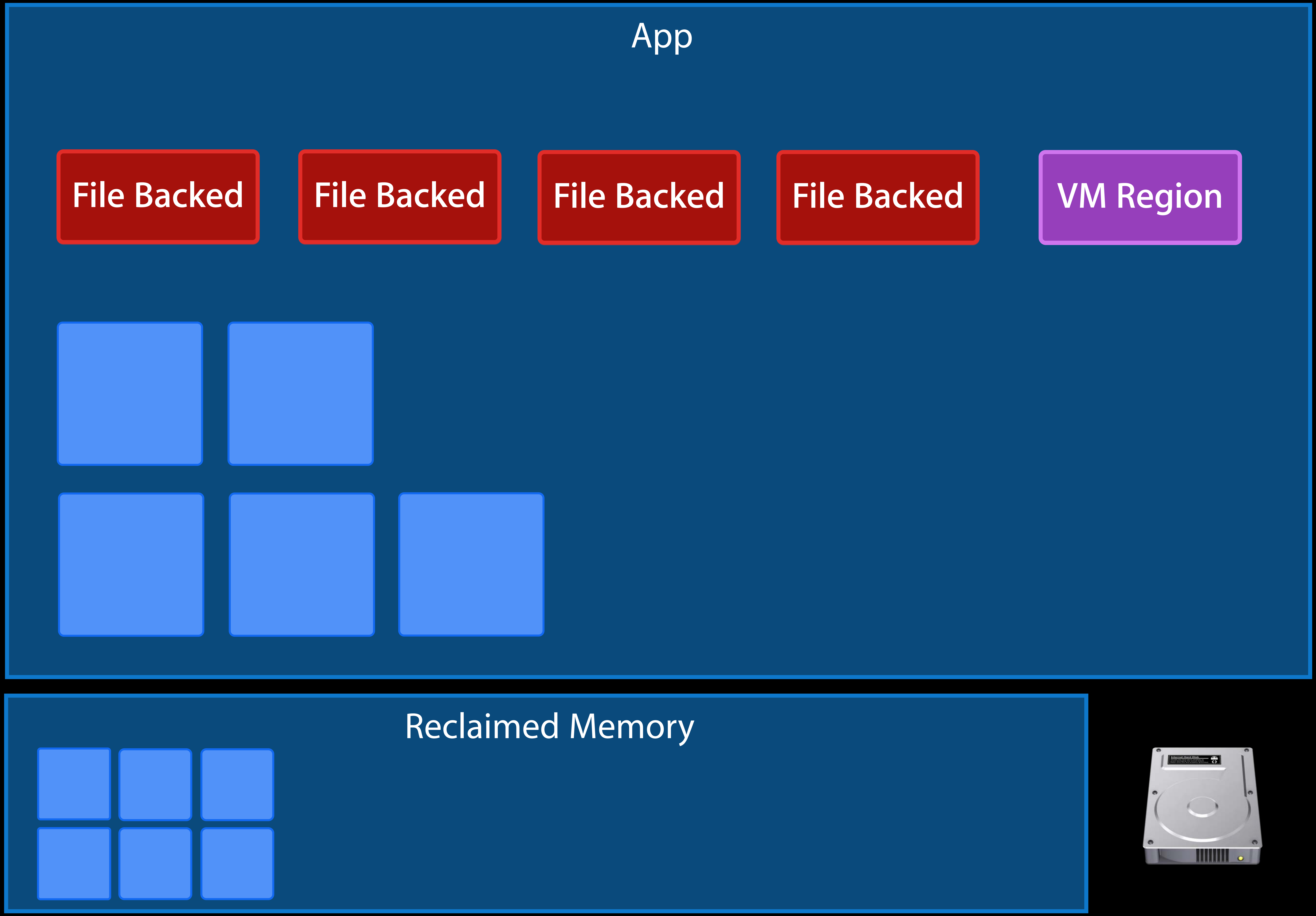
# Satisfying Demand for New Pages



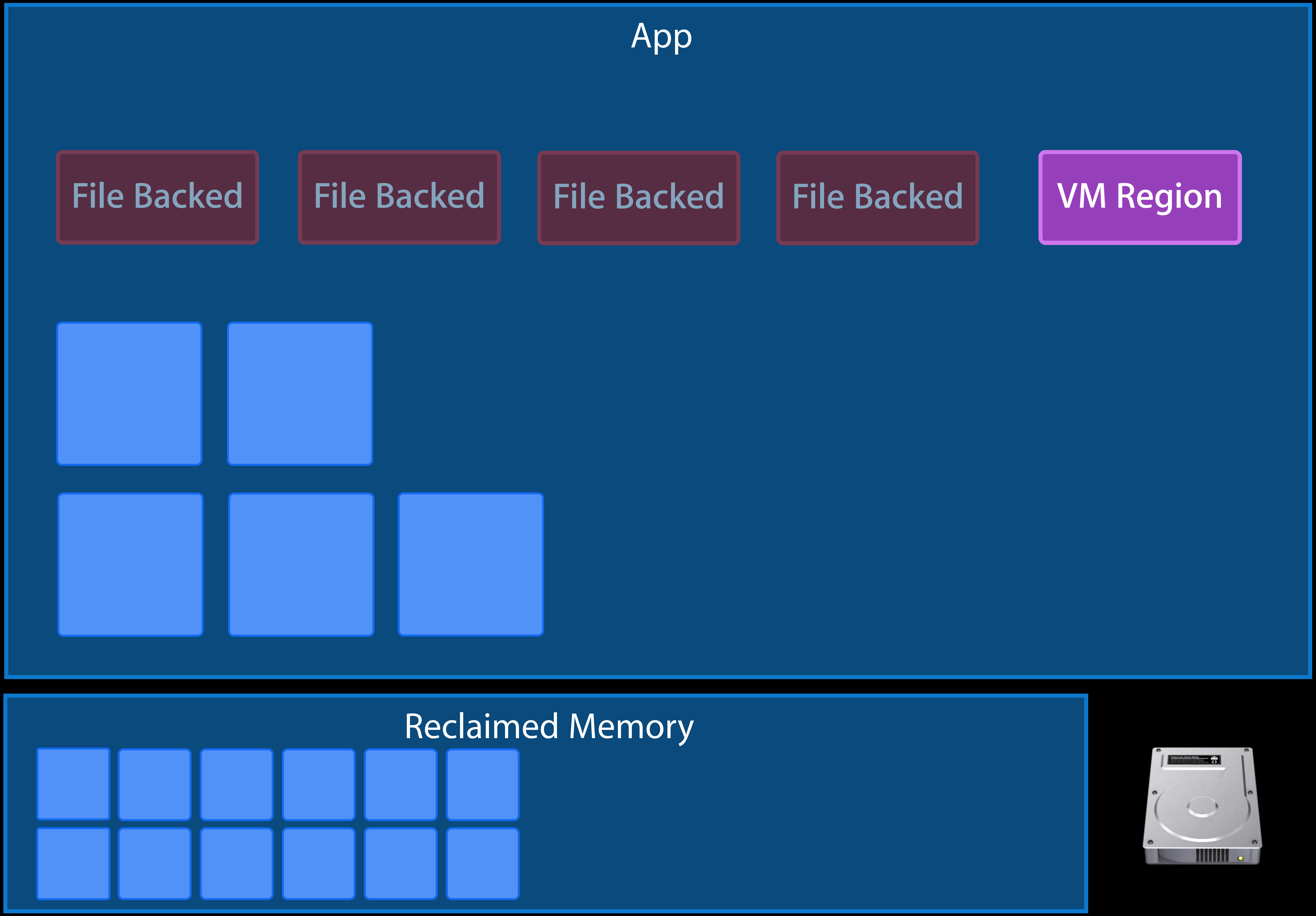
# Satisfying Demand for New Pages



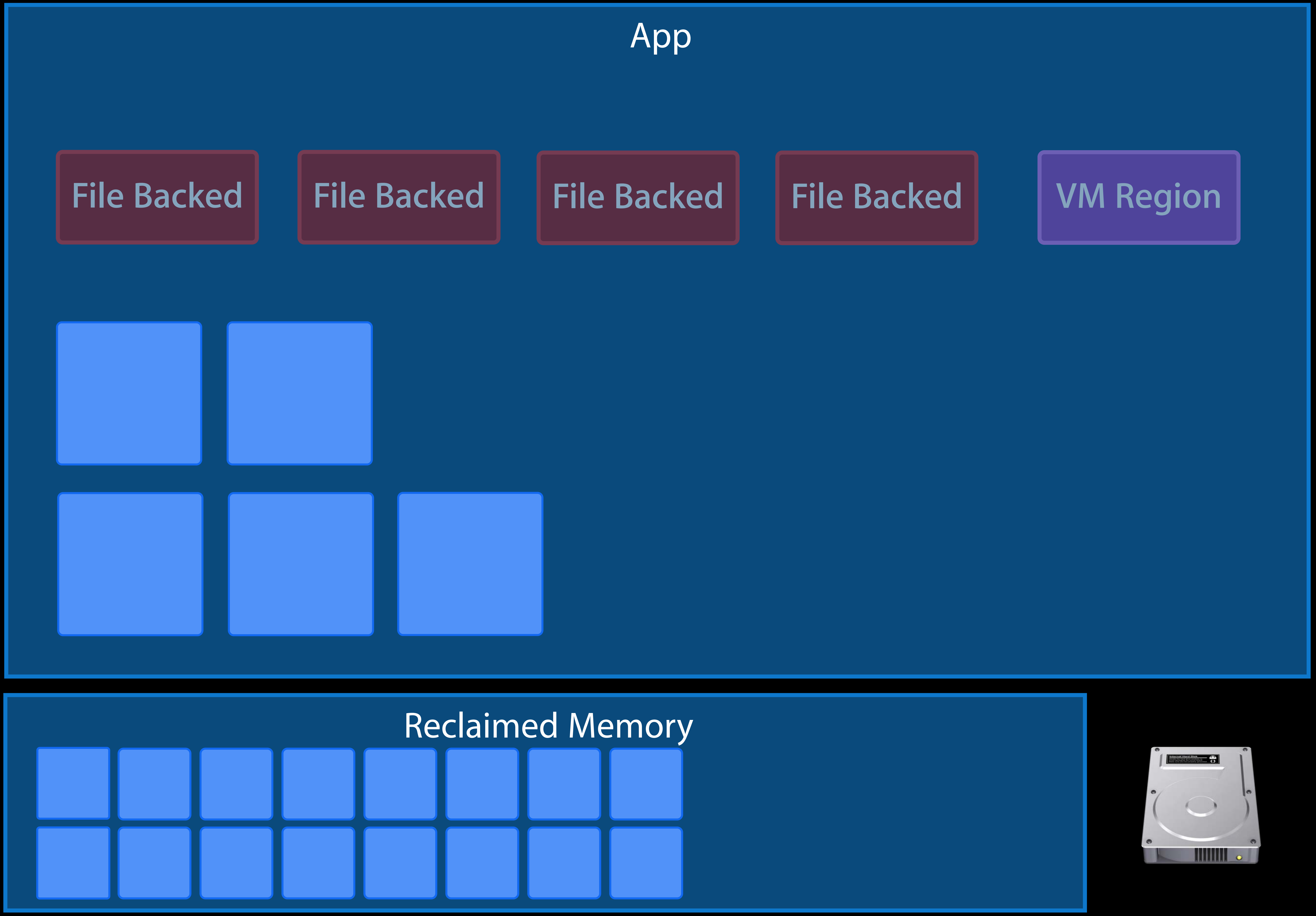
# Satisfying Demand for New Pages



# Satisfying Demand for New Pages



# Satisfying Demand for New Pages



App

File Backed

File Backed

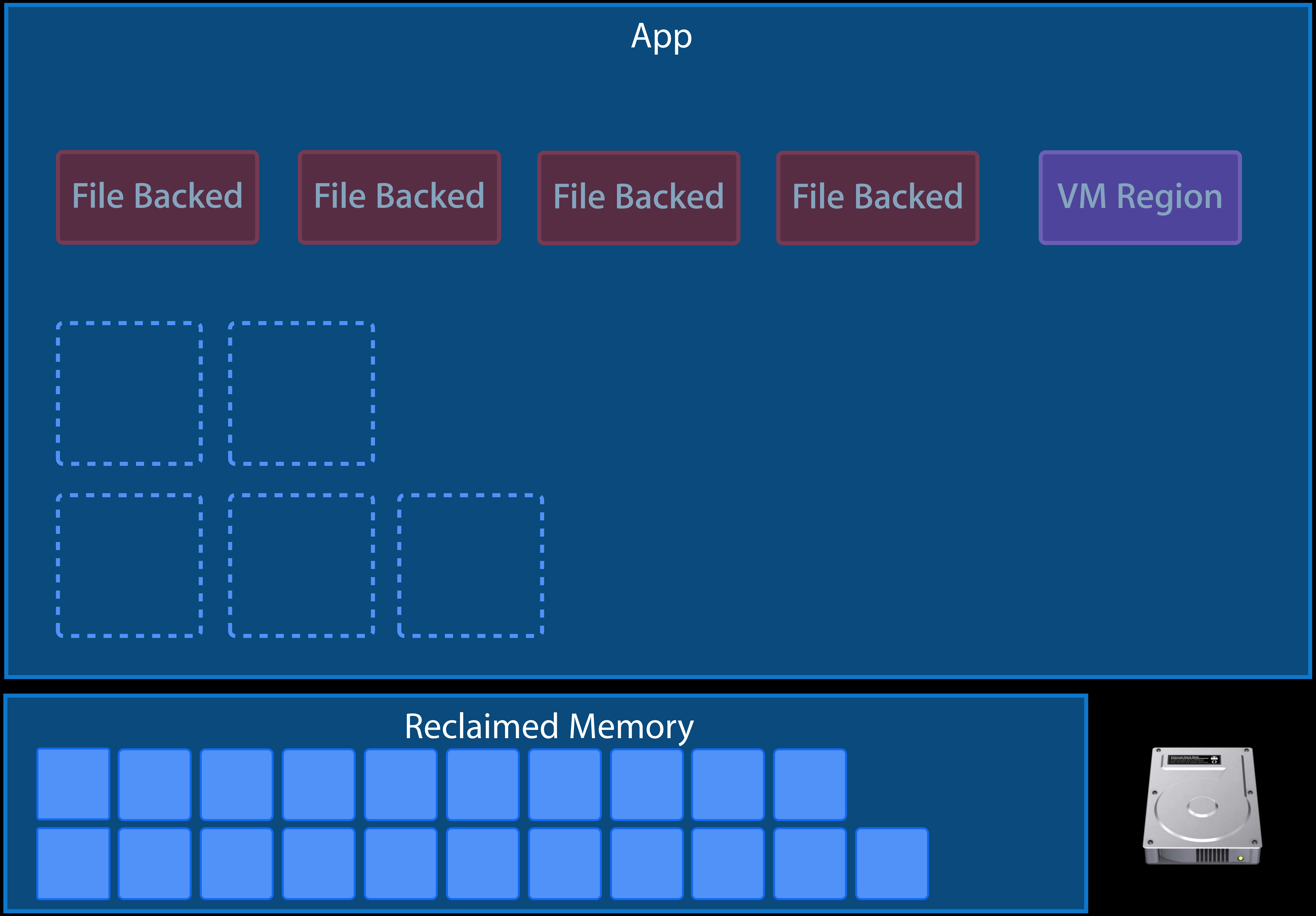
File Backed

File Backed

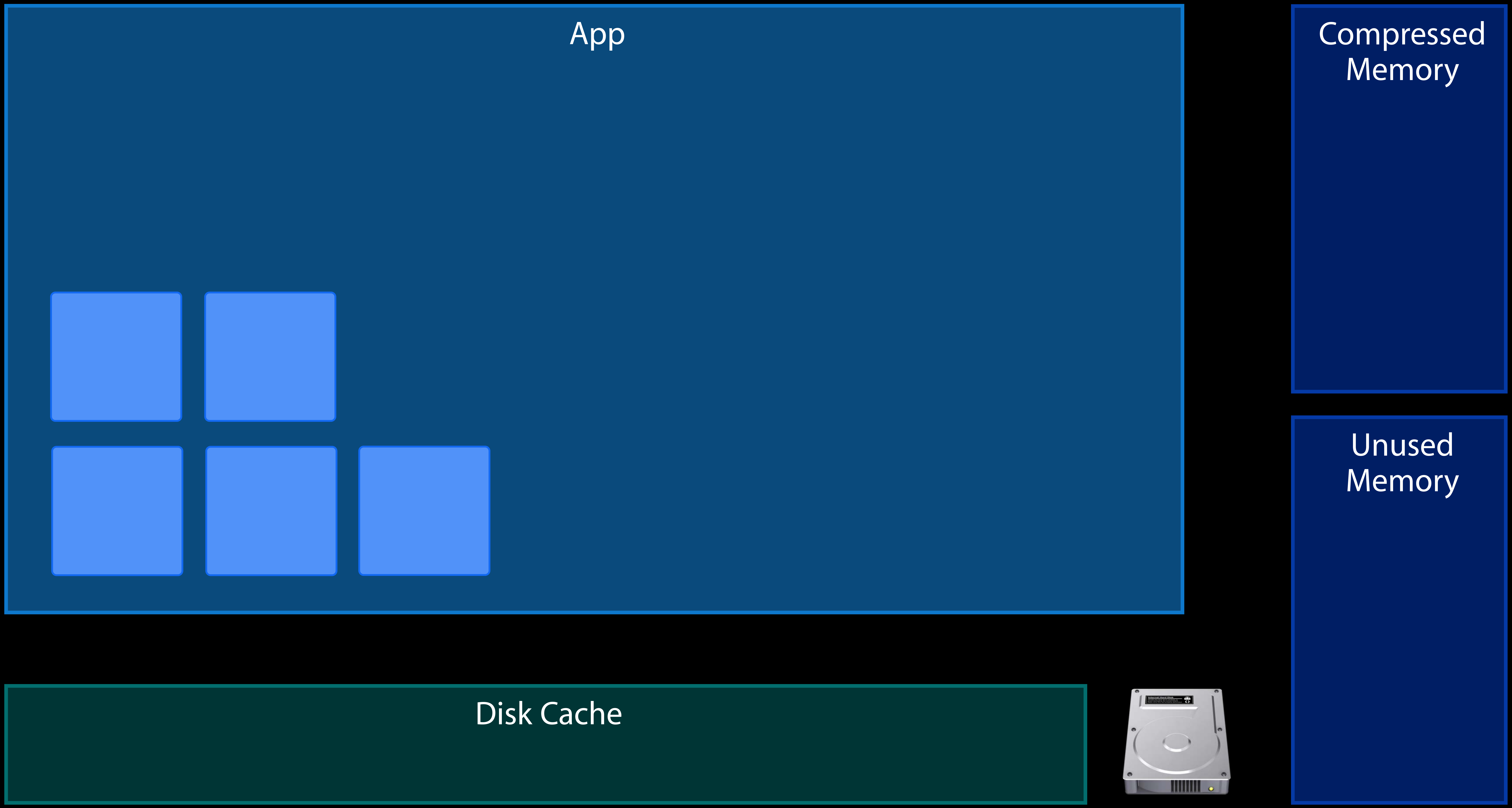
VM Region

Reclaimed Memory

# Satisfying Demand for New Pages



# Compressed Memory



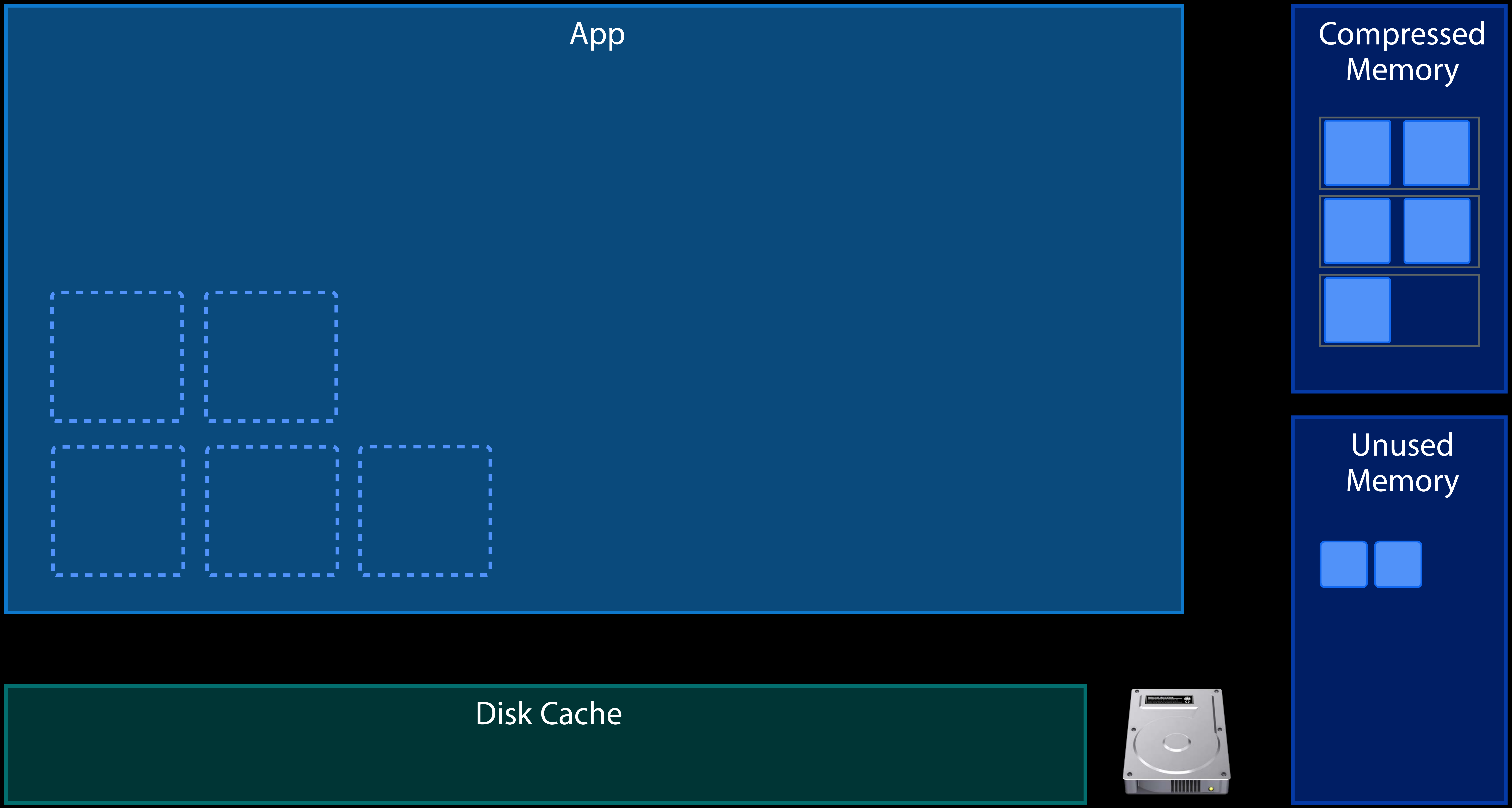
App

Compressed Memory

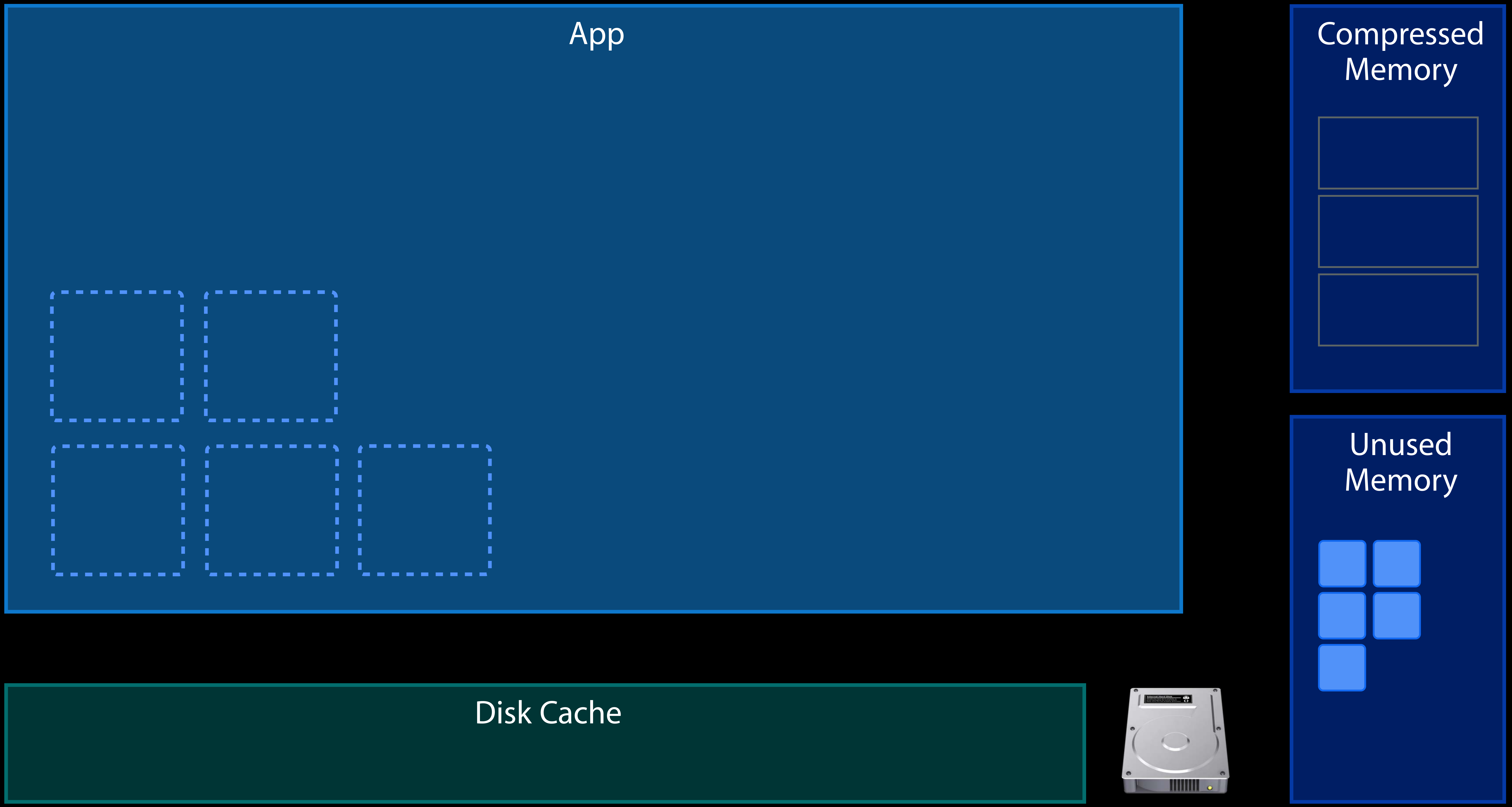
Unused Memory

Disk Cache

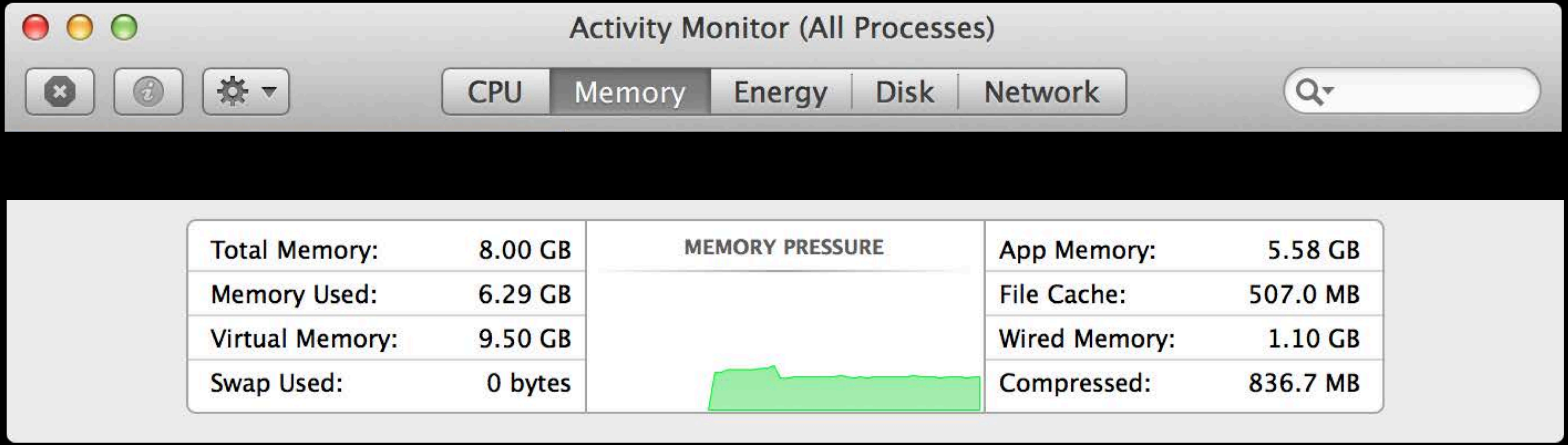
# Compressed Memory



# Compressed Memory



# Understanding System-Wide Behavior



| | | | |
|---|---|---|---|
| Total Memory: | 8.00 GB | App Memory: | 5.58 GB |
| Memory Used: | 6.29 GB | File Cache: | 507.0 MB |
| Virtual Memory: | 9.50 GB | Wired Memory: | 1.10 GB |
| Swap Used: | 0 bytes | Compressed: | 836.7 MB |

# Diving Deeper

# Diving Deeper

```
$ vm_stat 1
file-backed anonymous  cmprssed  cmprssor dcomprs   comprs pageins pageout swapins swapouts
     121820   1872398  1732648  1188839  681660 2878470  801581  210428  397327   567106
     121689   1872716  1732322  1188807     324       0      42       0       0        0
     122080   1871050  1732258  1188807      25       0       6       0       0        0
     121610   1873003  1731976  1188743     282       0      29       0       0        0
     121861   1872084  1731699  1188672     277       0      56       0      57        0
```

# Diving Deeper

```
$ vm_stat 1
file-backed anonymous cmprssed cmprssor dcomprs  comprs pageins pageout swapins swapouts
     121820   1872398  1732648  1188839  681660 2878470  801581  210428  397327   567106
     121689   1872716  1732322  1188807     324       0      42       0       0        0
     122080   1871050  1732258  1188807      25       0       6       0       0        0
     121610   1873003  1731976  1188743     282       0      29       0       0        0
     121861   1872084  1731699  1188672     277       0      56       0      57        0

$ vm_stat
File-backed pages:                       110808.
Anonymous pages:                        1775867.
Pages stored in compressor:             1838900.
Pages occupied by compressor:           1155000.
Decompressions:                          701085.
Compressions:                           3011761.
Pageins:                                 815922.
Pageouts:                                216464.
Swapins:                                 401147.
Swapouts:                                656148.
```

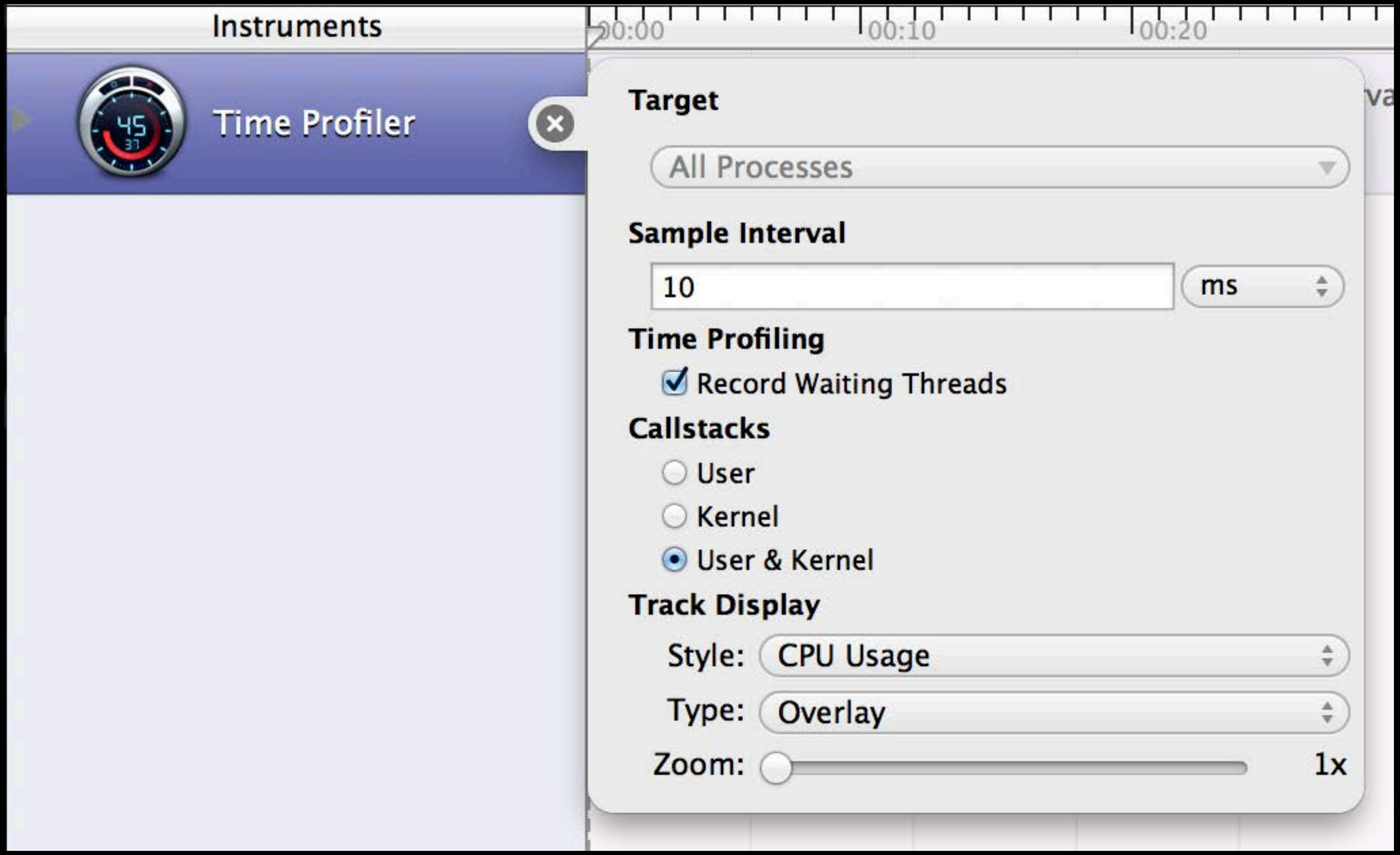
# Diving Deeper

```
$ vm_stat 1
file-backed anonymous cmprssed cmprssor dcomprs  comprs pageins pageout swapins swapouts
     121820   1872398  1732648  1188839  681660 2878470  801581  210428  397327   567106
     121689   1872716  1732322  1188807     324       0      42       0       0        0
     122080   1871050  1732258  1188807      25       0       6       0       0        0
     121610   1873003  1731976  1188743     282       0      29       0       0        0
     121861   1872084  1731699  1188672     277       0      56       0      57        0

$ vm_stat
File-backed pages:                       110808.
Anonymous pages:                        1775867.
Pages stored in compressor:             1838900.
Pages occupied by compressor:           1155000.
Decompressions:                          701085.
Compressions:                           3011761.
Pageins:                                 815922.
Pageouts:                                216464.
Swapins:                                 401147.
Swapouts:                                656148.
```

# Diving Deeper

```
$ vm_stat 1
file-backed anonymous  cmprssed  cmprssor dcomprs   comprs  pageins  pageout  swapins  swapouts
     121820   1872398   1732648   1188839  681660  2878470   801581   210428   397327    567106
     121689   1872716   1732322   1188807     324        0       42        0        0         0
     122080   1871050   1732258   1188807      25        0        6        0        0         0
     121610   1873003   1731976   1188743     282        0       29        0        0         0
     121861   1872084   1731699   1188672     277        0       56        0       57         0

$ vm_stat
File-backed pages:                        110808.
Anonymous pages:                         1775867.
Pages stored in compressor:              1838900.
Pages occupied by compressor:            1155000.
Decompressions:                           701085.
Compressions:                            3011761.
Pageins:                                  815922.
Pageouts:                                 216464.
Swapins:                                  401147.
Swapouts:                                 656148.
```

# Detecting Swapping in Your App



**Target**

All Processes

**Sample Interval**

10 ms

**Time Profiling**

☑ Record Waiting Threads

**Callstacks**

○ User

○ Kernel

● User & Kernel

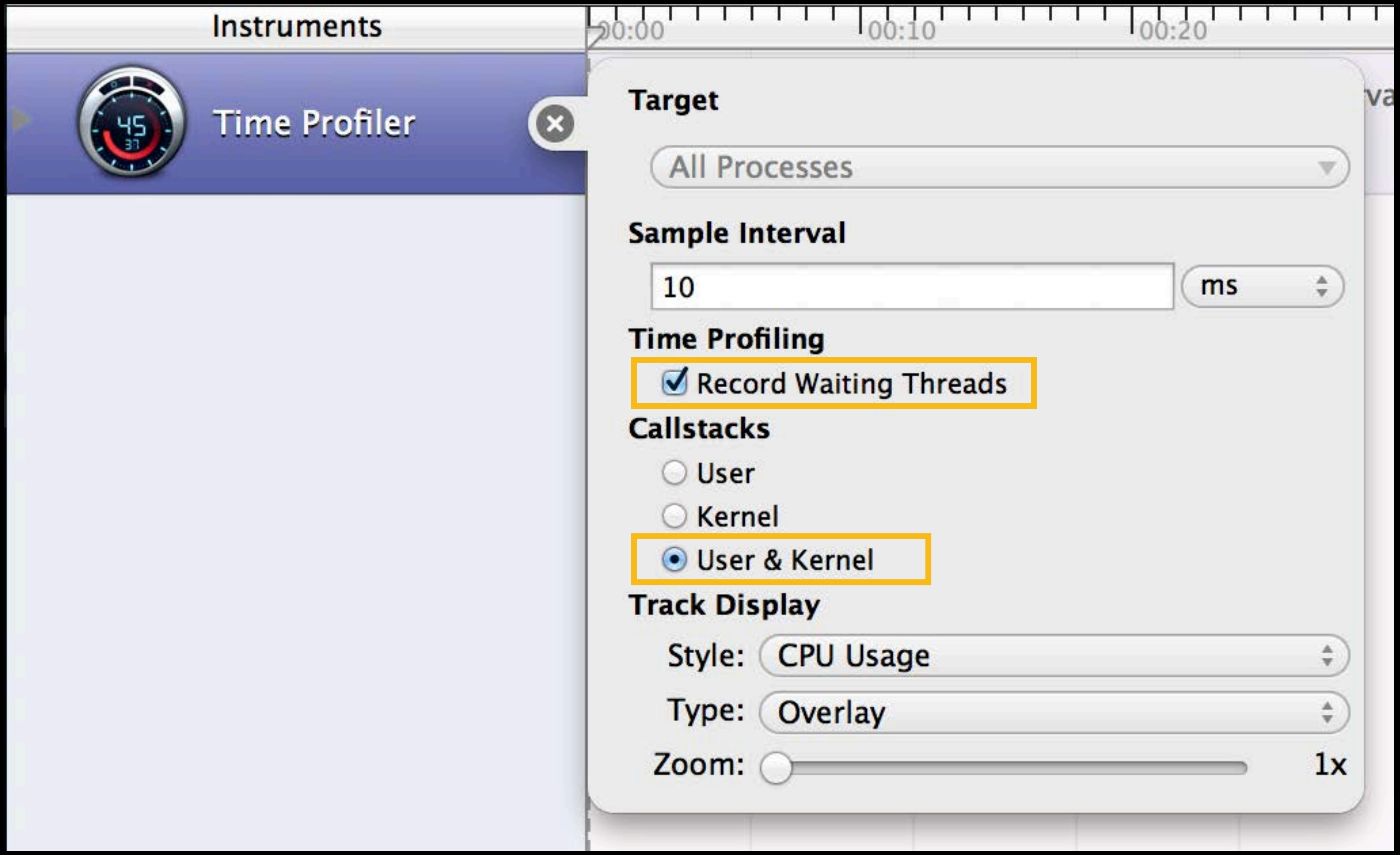
**Track Display**

Style: CPU Usage

Type: Overlay

Zoom: 1x

# Detecting Swapping in Your App



**Target**

All Processes

**Sample Interval**

10 ms

**Time Profiling**

☑ Record Waiting Threads

**Callstacks**

○ User

○ Kernel

● User & Kernel

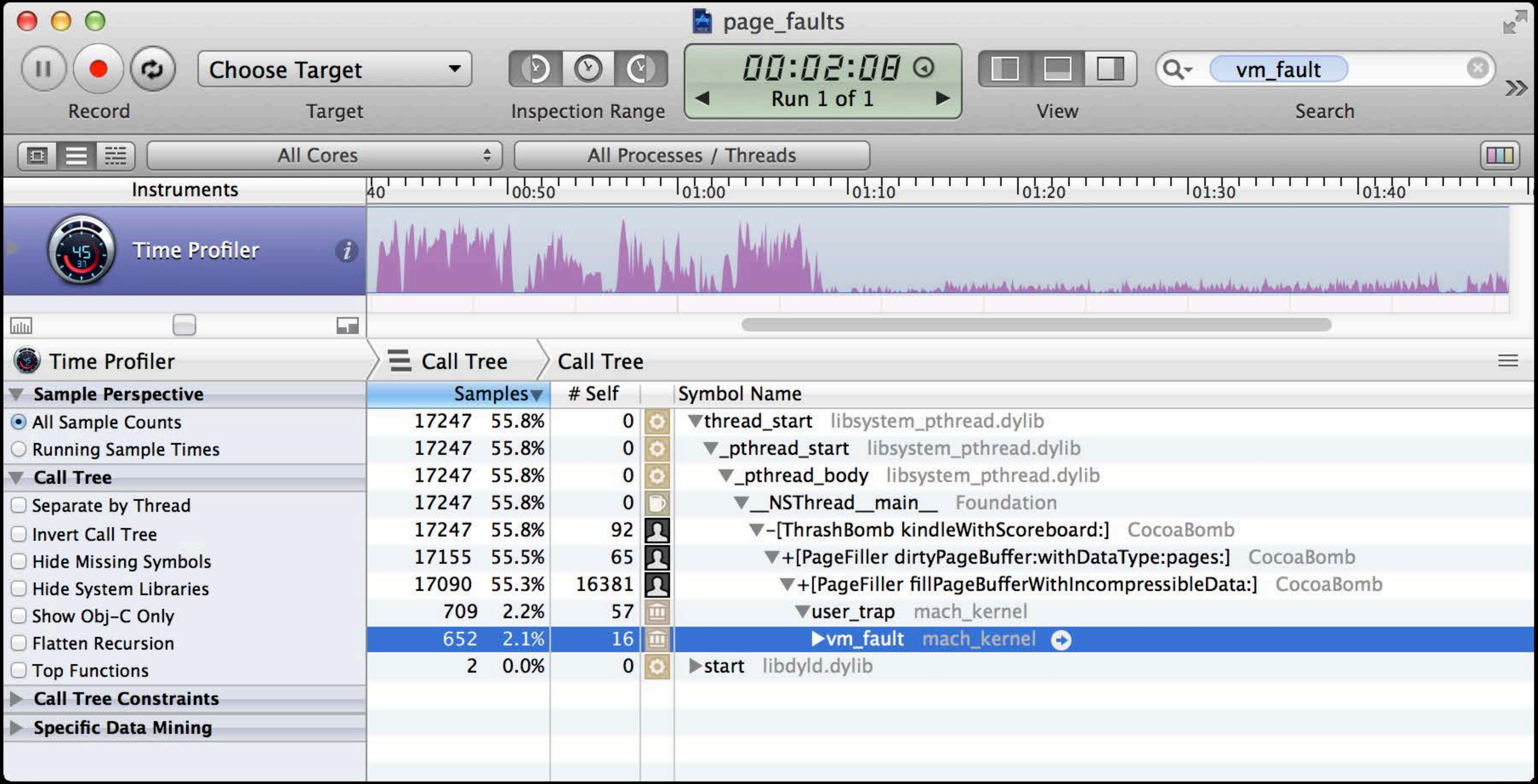
**Track Display**

Style: CPU Usage

Type: Overlay

Zoom: 1x

# Detecting Swapping in Your App



**Sample Perspective**

- ● All Sample Counts
- ○ Running Sample Times

**Call Tree**

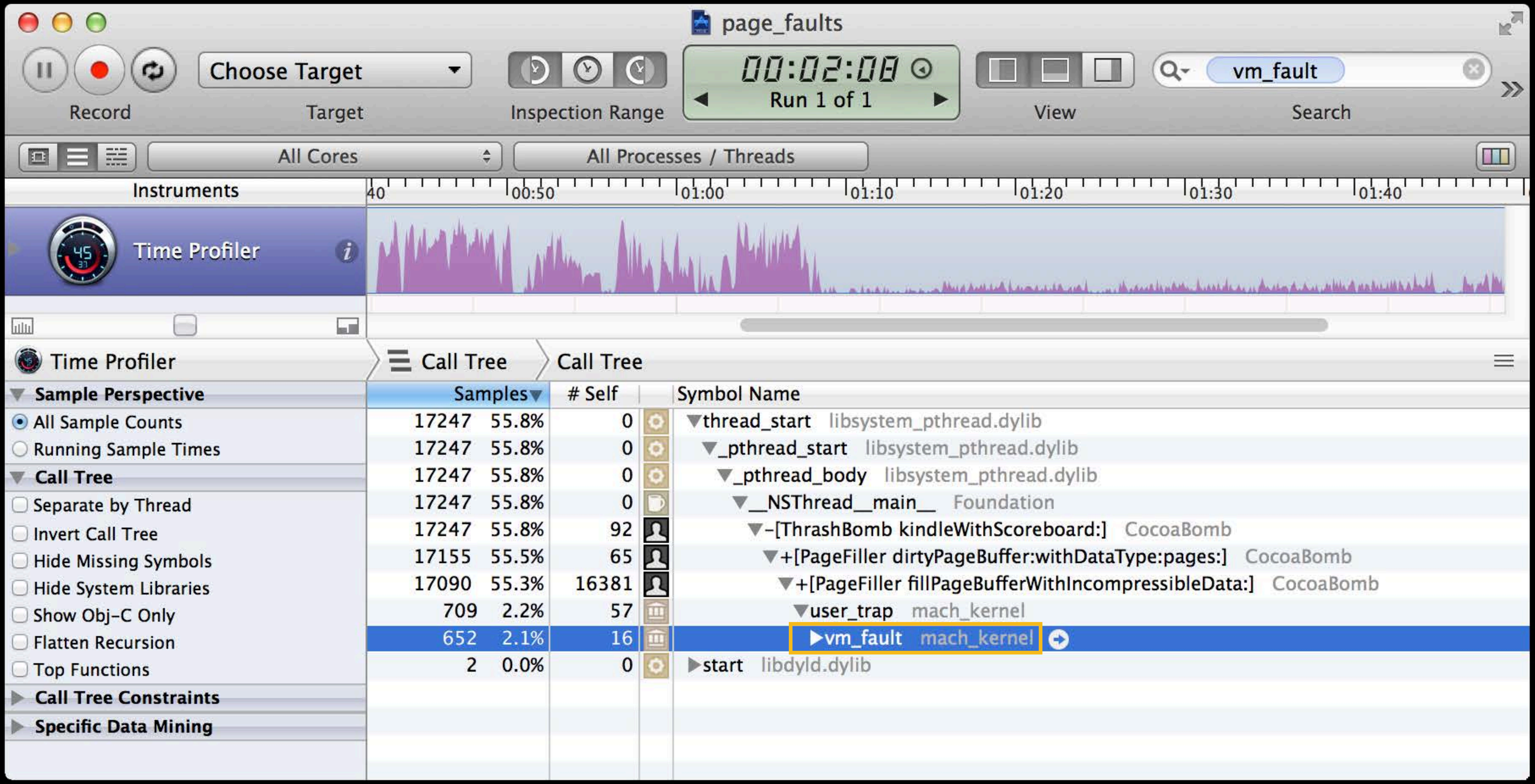
- □ Separate by Thread
- □ Invert Call Tree
- □ Hide Missing Symbols
- □ Hide System Libraries
- □ Show Obj-C Only
- □ Flatten Recursion
- □ Top Functions

**Call Tree Constraints**

**Specific Data Mining**

| Samples | | # Self | Symbol Name |
|---|---|---|---|
| 17247 | 55.8% | 0 | ▼thread_start libsystem_pthread.dylib |
| 17247 | 55.8% | 0 | ▼_pthread_start libsystem_pthread.dylib |
| 17247 | 55.8% | 0 | ▼_pthread_body libsystem_pthread.dylib |
| 17247 | 55.8% | 0 | ▼__NSThread__main__ Foundation |
| 17247 | 55.8% | 92 | ▼-[ThrashBomb kindleWithScoreboard:] CocoaBomb |
| 17155 | 55.5% | 65 | ▼+[PageFiller dirtyPageBuffer:withDataType:pages:] CocoaBomb |
| 17090 | 55.3% | 16381 | ▼+[PageFiller fillPageBufferWithIncompressibleData:] CocoaBomb |
| 709 | 2.2% | 57 | ▼user_trap mach_kernel |
| 652 | 2.1% | 16 | ▶vm_fault mach_kernel |
| 2 | 0.0% | 0 | ▶start libdyld.dylib |

# Detecting Swapping in Your App



▼ Sample Perspective

- ● All Sample Counts
- ○ Running Sample Times

▼ Call Tree

- □ Separate by Thread
- □ Invert Call Tree
- □ Hide Missing Symbols
- □ Hide System Libraries
- □ Show Obj-C Only
- □ Flatten Recursion
- □ Top Functions

▶ Call Tree Constraints

▶ Specific Data Mining

| Samples▼ | | # Self | Symbol Name |
|---|---|---|---|
| 17247 | 55.8% | 0 | ▼thread_start libsystem_pthread.dylib |
| 17247 | 55.8% | 0 | ▼_pthread_start libsystem_pthread.dylib |
| 17247 | 55.8% | 0 | ▼_pthread_body libsystem_pthread.dylib |
| 17247 | 55.8% | 0 | ▼__NSThread__main__ Foundation |
| 17247 | 55.8% | 92 | ▼-[ThrashBomb kindleWithScoreboard:] CocoaBomb |
| 17155 | 55.5% | 65 | ▼+[PageFiller dirtyPageBuffer:withDataType:pages:] CocoaBomb |
| 17090 | 55.3% | 16381 | ▼+[PageFiller fillPageBufferWithIncompressibleData:] CocoaBomb |
| 709 | 2.2% | 57 | ▼user_trap mach_kernel |
| 652 | 2.1% | 16 | ▶vm_fault mach_kernel |
| 2 | 0.0% | 0 | ▶start libdyld.dylib |

# Collecting Data

- `sudo sysdiagnose <AppName>`
- Produces e.g. /var/tmp/sysdiagnose_2013.06.04_19-36-02-PDT_481.tar.gz
  - spindump – Time Profiler style sampling
  - heap
  - leaks
  - footprint
  - vm_stat
  - fs_usage
  - and much more!
- Can also be triggered with shift-control-option-command-period

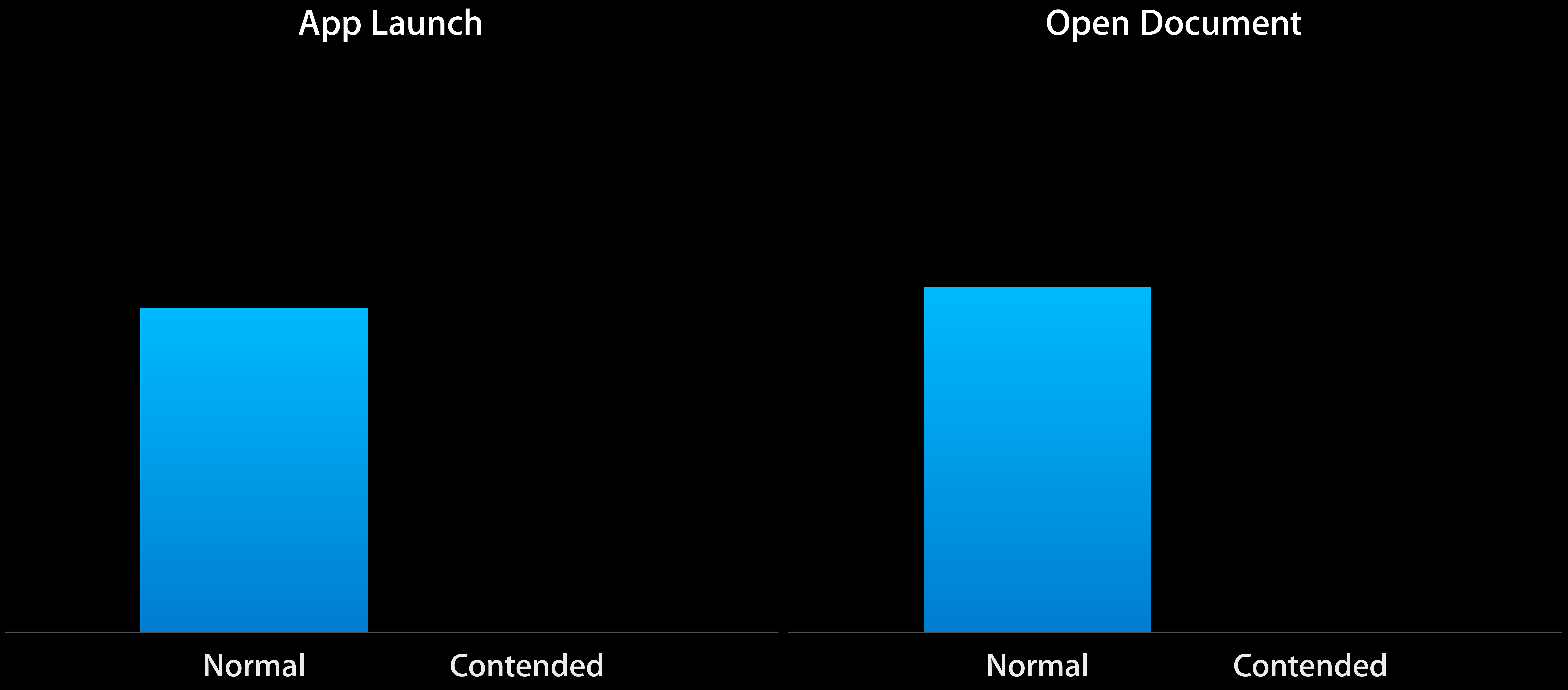
# Memory Recap

- Pay attention to the entire footprint of your app
- When trying to reduce your memory usage:
  - Check for leaks and heap growth
  - Check for unneeded VM regions
  - Check for duplicated memory
- Adopt purgeable memory or NSCache
- Bigger apps are more likely to slow down under memory pressure

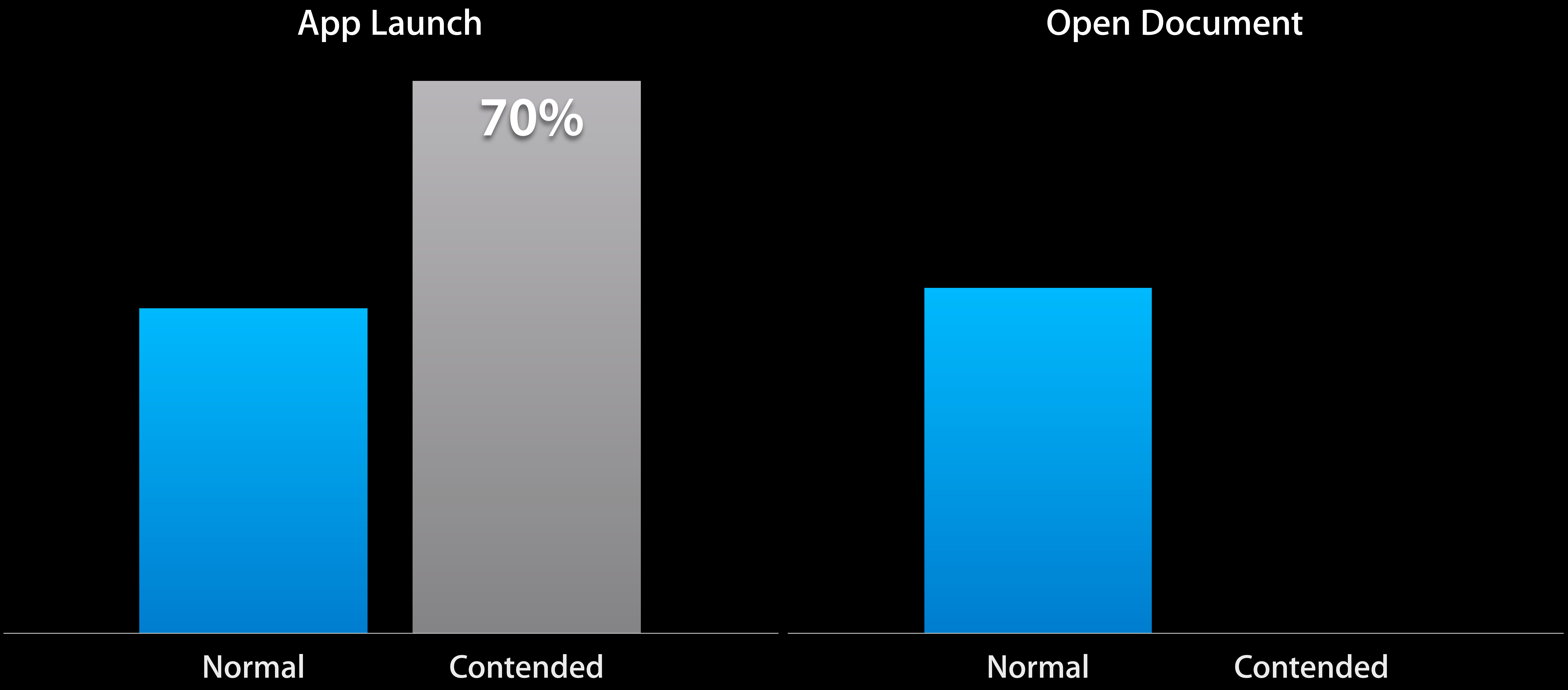
# Disk IO

# Importance of IO Performance

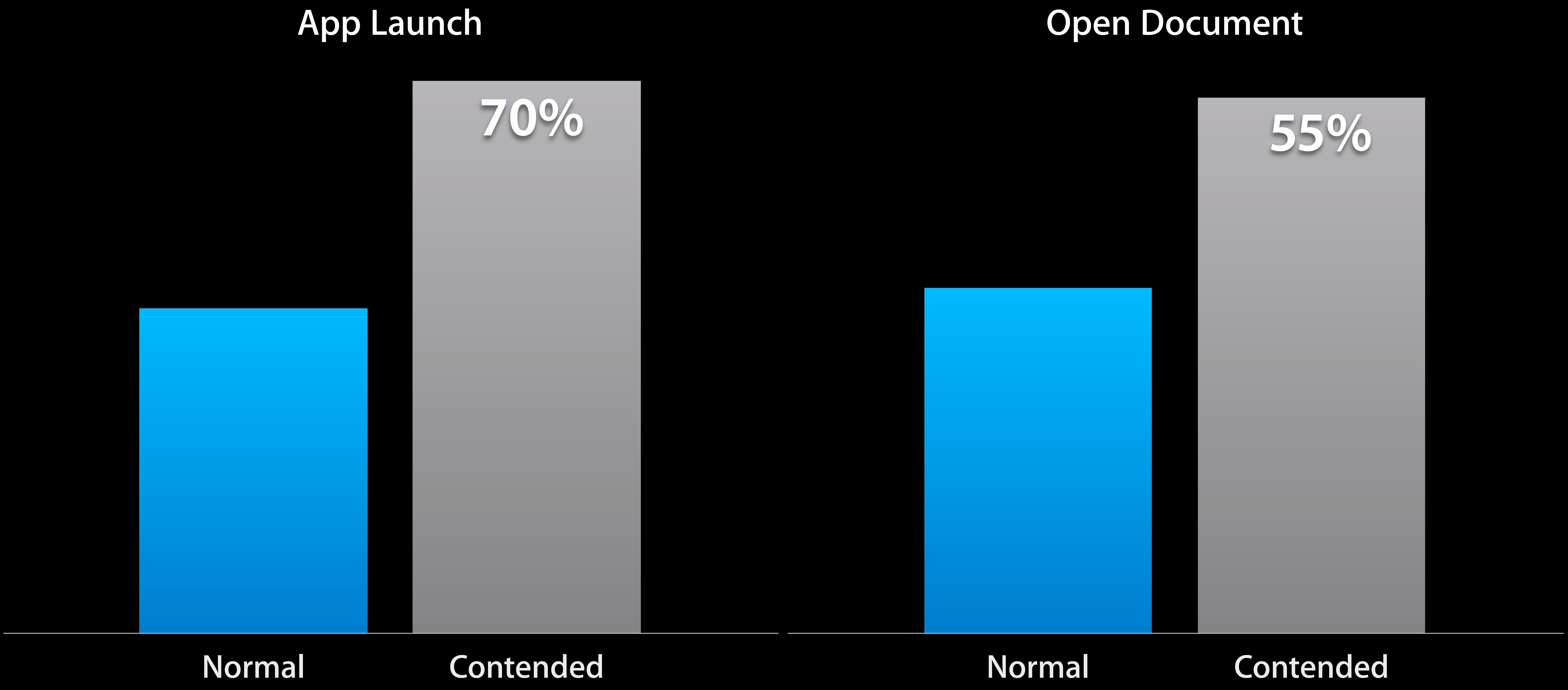
# Importance of IO Performance



# Importance of IO Performance



# Importance of IO Performance

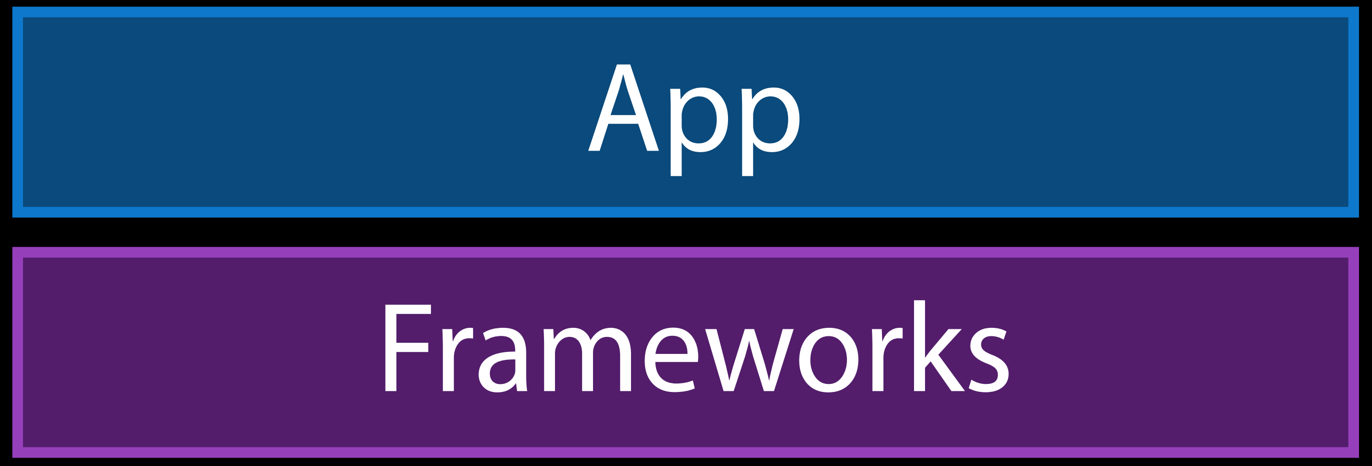


# Storage Stack

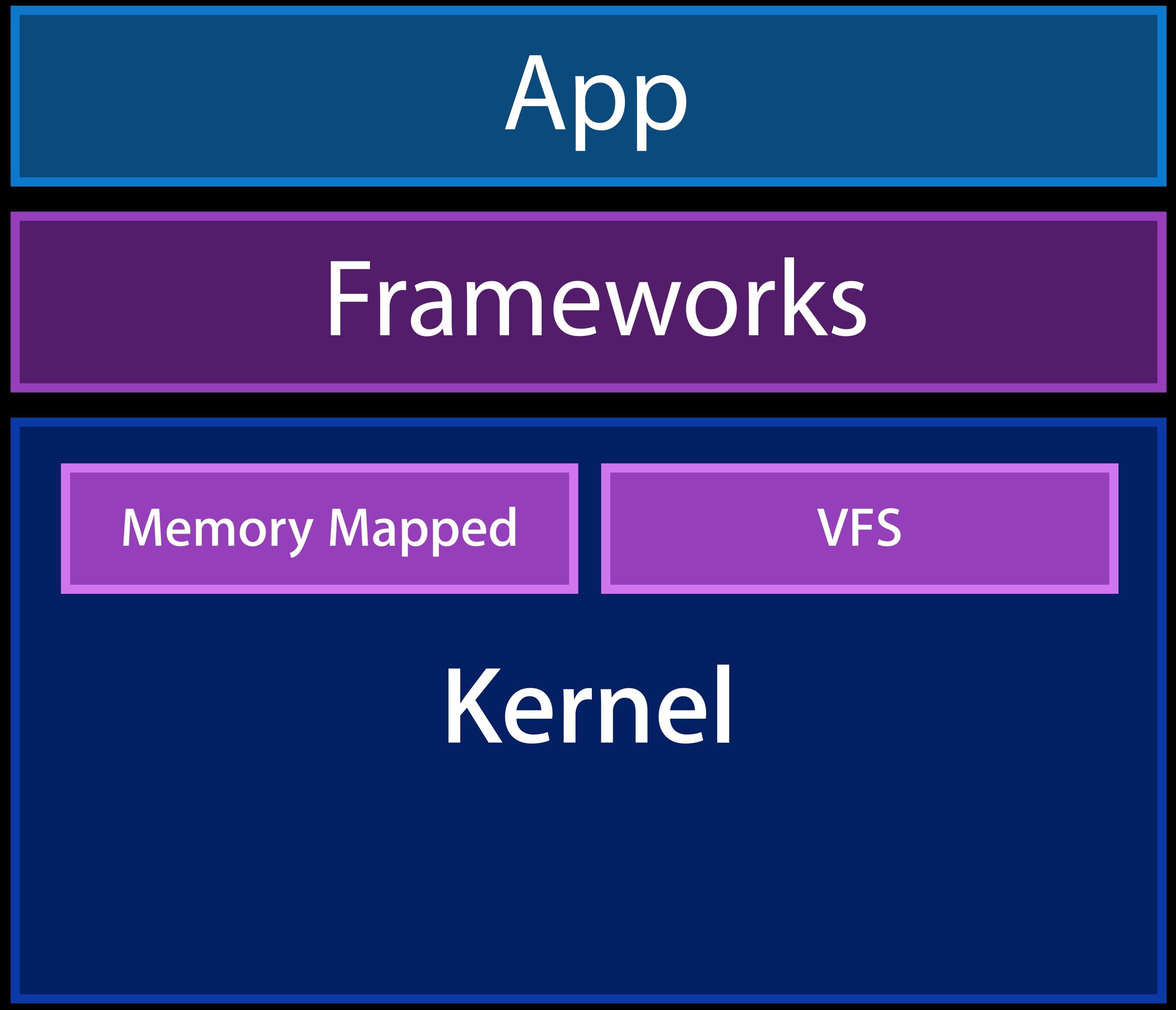
# Storage Stack

App

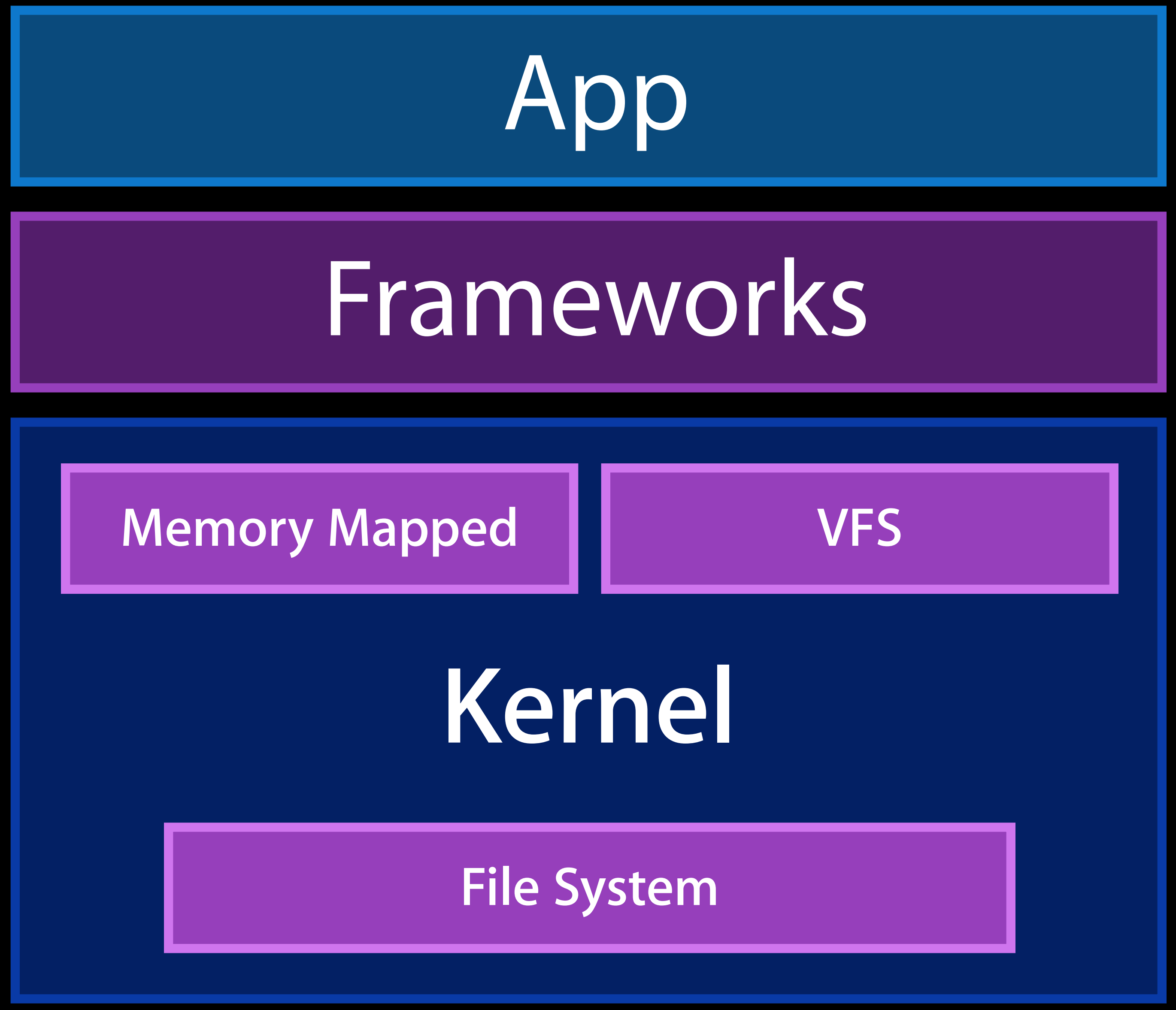
# Storage Stack



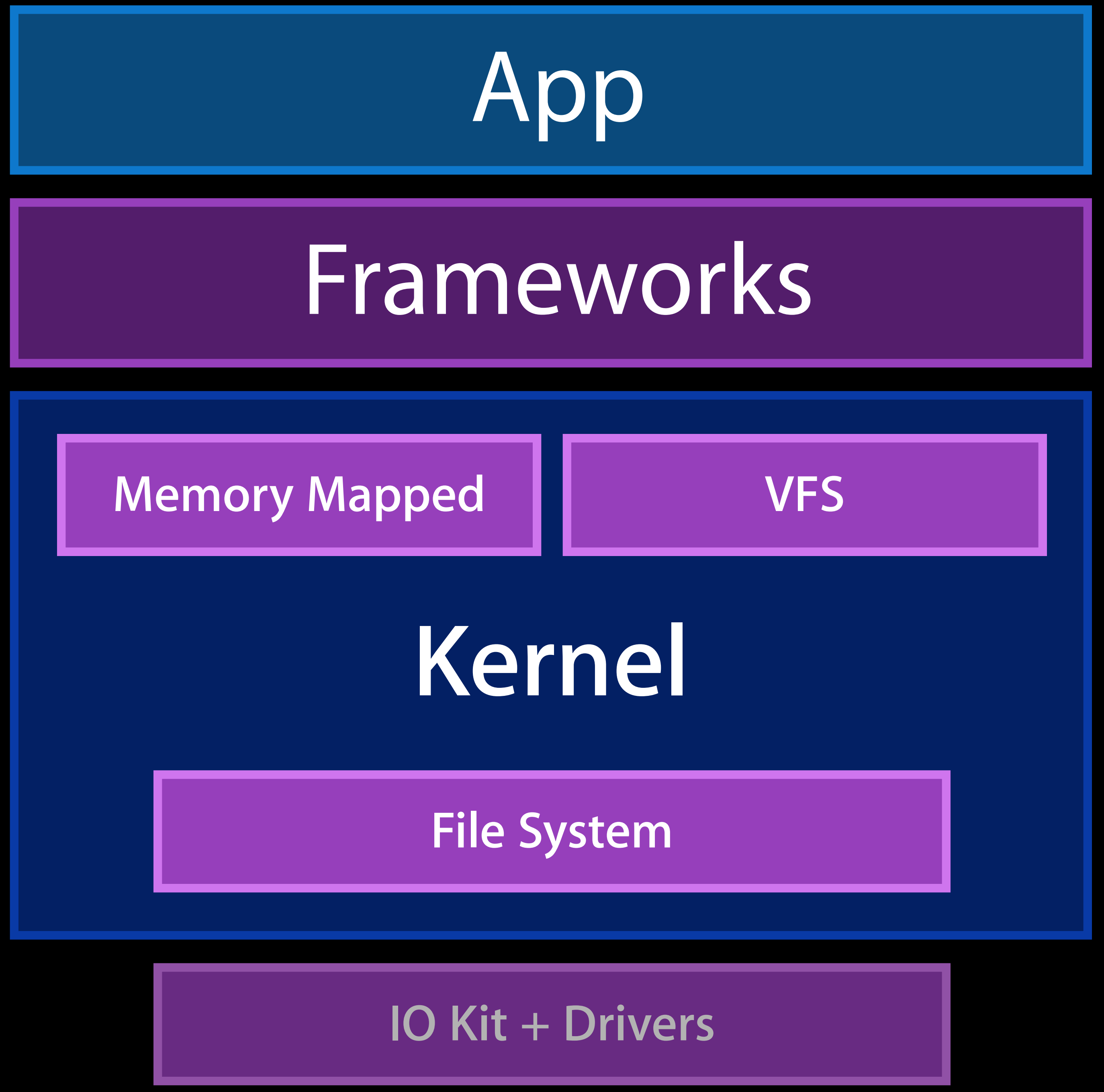
# Storage Stack



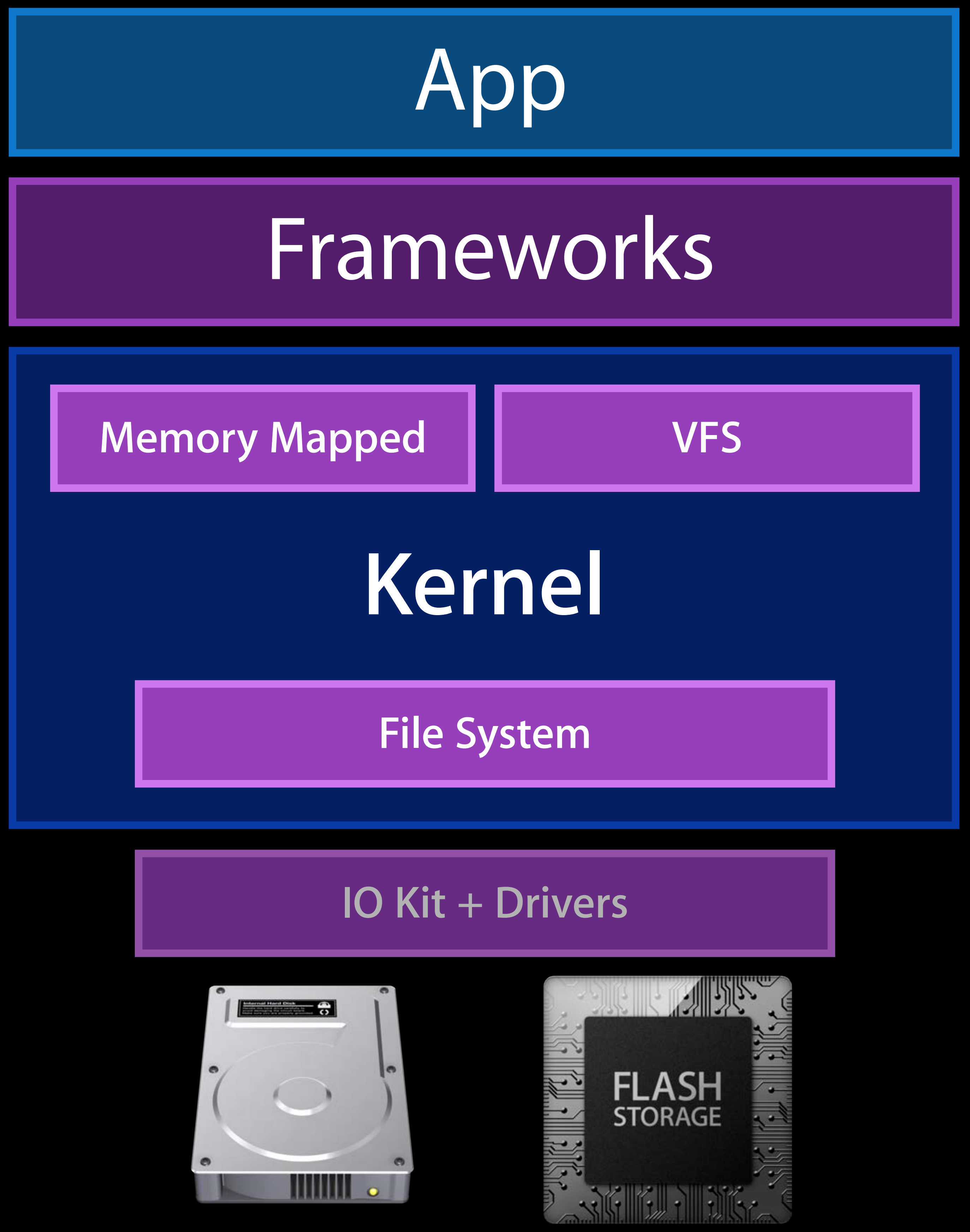
# Storage Stack



# Storage Stack



# Storage Stack



App

Frameworks

Memory Mapped

VFS

Kernel

File System

IO Kit + Drivers

# Storage Devices

## Consider Both

Performance numbers approximate and not representative of any specific product.

# Storage Devices

## Consider Both

SSD

HDD

Performance numbers approximate and not representative of any specific product.

# Storage Devices

## Consider Both

| | SSD | HDD |
|---|---|---|
| Seek Penalty | None | 10ms |
| IOs per Second | 3k-30k IOPS | 80 IOPS |
| Sequential Speed | 400 MB/s | 160 MB/s |

Performance numbers approximate and not representative of any specific product.

# Storage Devices

## Consider Both

| | SSD | HDD |
|---|---|---|
| Seek Penalty | None | 10ms |
| IOs per Second | 3k-30k IOPS | 80 IOPS |
| Sequential Speed | 400 MB/s | 160 MB/s |
| Parallelism | Limited | None |

Performance numbers approximate and not representative of any specific product.

# Storage Devices

## Consider Both

| | SSD | HDD |
|---|---|---|
| Seek Penalty | None | 10ms |
| IOs per Second | 3k-30k IOPS | 80 IOPS |
| Sequential Speed | 400 MB/s | 160 MB/s |
| Parallelism | Limited | None |
| Read versus Write | Writes more expensive | Symmetric |

Performance numbers approximate and not representative of any specific product.

# High-Performance IO Is Difficult

- Avoid causing thrashing on HDDs
- Keep queue filled for SSDs
- Use appropriate buffer sizes
- Compute on data concurrently with IO
- Avoid copying data unnecessarily

# Maximize IO Performance

## Let dispatch IO handle doing IO the fastest way

- Part of Grand Central Dispatch
- Available since OS X 10.7
- Declarative API for file access
- Encapsulates best-practices

# Dispatch IO

## Processing a large file

```
dispatch_queue_t queue = dispatch_queue_create("com.example.FileProcessing", NULL);
dispatch_io_t io = dispatch_io_create_with_path(DISPATCH_IO_RANDOM, path,
     O_RDONLY, 0, queue , NULL);
dispatch_io_set_high_water(io, 32 * 1024);
dispatch_io_read(io, 0, SIZE_MAX, queue,
     ^(bool done, dispatch_data_t data, int error){
         if (error == 0)
             dispatch_data_apply(data, ^(rgn, offset, ptr, len){
                 /* process len bytes at ptr */
             });
     }
});
```

# Dispatch IO

## Processing a large file

```
dispatch_queue_t queue = dispatch_queue_create("com.example.FileProcessing", NULL);
dispatch_io_t io = dispatch_io_create_with_path(DISPATCH_IO_RANDOM, path,
     O_RDONLY, 0, queue , NULL);
dispatch_io_set_high_water(io, 32 * 1024);
dispatch_io_read(io, 0, SIZE_MAX, queue,
     ^(bool done, dispatch_data_t data, int error){
         if (error == 0)
             dispatch_data_apply(data, ^(rgn, offset, ptr, len){
                 /* process len bytes at ptr */
             });
     }
});
```

# Dispatch IO

## Processing a large file

```
dispatch_queue_t queue = dispatch_queue_create("com.example.FileProcessing", NULL);
dispatch_io_t io = dispatch_io_create_with_path(DISPATCH_IO_RANDOM, path,
     O_RDONLY, 0, queue , NULL);
dispatch_io_set_high_water(io, 32 * 1024);
dispatch_io_read(io, 0, SIZE_MAX, queue,
     ^(bool done, dispatch_data_t data, int error){
         if (error == 0)
             dispatch_data_apply(data, ^(rgn, offset, ptr, len){
                 /* process len bytes at ptr */
             });
     }
});
```

# Dispatch IO

## Processing a large file

```
dispatch_queue_t queue = dispatch_queue_create("com.example.FileProcessing", NULL);
dispatch_io_t io = dispatch_io_create_with_path(DISPATCH_IO_RANDOM, path,
     O_RDONLY, 0, queue , NULL);
dispatch_io_set_high_water(io, 32 * 1024);
dispatch_io_read(io, 0, SIZE_MAX, queue,
     ^(bool done, dispatch_data_t data, int error){
         if (error == 0)
             dispatch_data_apply(data, ^(rgn, offset, ptr, len){
                 /* process len bytes at ptr */
             });
     }
});
```

# Dispatch IO

## Processing a large file

```
dispatch_queue_t queue = dispatch_queue_create("com.example.FileProcessing", NULL);
dispatch_io_t io = dispatch_io_create_with_path(DISPATCH_IO_RANDOM, path,
     O_RDONLY, 0, queue , NULL);
dispatch_io_set_high_water(io, 32 * 1024);
dispatch_io_read(io, 0, SIZE_MAX, queue,
     ^(bool done, dispatch_data_t data, int error){
         if (error == 0)
             dispatch_data_apply(data, ^(rgn, offset, ptr, len){
                 /* process len bytes at ptr */
             });
     }
});
```

# Dispatch IO

## Processing a large file

```
dispatch_queue_t queue = dispatch_queue_create("com.example.FileProcessing", NULL);
dispatch_io_t io = dispatch_io_create_with_path(DISPATCH_IO_RANDOM, path,
     O_RDONLY, 0, queue , NULL);
dispatch_io_set_high_water(io, 32 * 1024);
dispatch_io_read(io, 0, SIZE_MAX, queue,
     ^(bool done, dispatch_data_t data, int error){
         if (error == 0)
             dispatch_data_apply(data, ^(rgn, offset, ptr, len){
                 /* process len bytes at ptr */
             });
     }
});
```

# Dispatch IO

## Reading many files

```
dispatch_queue_t queue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_LOW,0);
for (NSString *path in imagePaths) {
    dispatch_io_t io = dispatch_io_create_with_path(DISPATCH_IO_RANDOM,
        [path fileSystemRepresentation], O_RDONLY, 0, queue , NULL);
    dispatch_io_set_low_water(io, SIZE_MAX);
    dispatch_io_read(io, 0, SIZE_MAX, queue,
        ^(bool done, dispatch_data_t data, int error){
            if (error == 0){
                NSImage *image = [[NSImage alloc] initWithData:(NSData*)data];
                @synchronized(images){ [images addObject:image]; }
            }
        });
}
```

# Dispatch IO

## Reading many files

```
dispatch_queue_t queue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_LOW,0);
for (NSString *path in imagePaths) {
    dispatch_io_t io = dispatch_io_create_with_path(DISPATCH_IO_RANDOM,
        [path fileSystemRepresentation], O_RDONLY, 0, queue , NULL);
    dispatch_io_set_low_water(io, SIZE_MAX);
    dispatch_io_read(io, 0, SIZE_MAX, queue,
        ^(bool done, dispatch_data_t data, int error){
            if (error == 0){
                NSImage *image = [[NSImage alloc] initWithData:(NSData*)data];
                @synchronized(images){ [images addObject:image]; }
            }
        });
}
```

# Dispatch IO

## Reading many files

```
dispatch_queue_t queue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_LOW,0);
for (NSString *path in imagePaths) {
    dispatch_io_t io = dispatch_io_create_with_path(DISPATCH_IO_RANDOM,
        [path fileSystemRepresentation], O_RDONLY, 0, queue , NULL);
    dispatch_io_set_low_water(io, SIZE_MAX);
    dispatch_io_read(io, 0, SIZE_MAX, queue,
        ^(bool done, dispatch_data_t data, int error){
            if (error == 0){
                NSImage *image = [[NSImage alloc] initWithData:(NSData*)data];
                @synchronized(images){ [images addObject:image]; }
            }
        });
}
```

# Dispatch IO

## Reading many files

```
dispatch_queue_t queue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_LOW,0);
for (NSString *path in imagePaths) {
    dispatch_io_t io = dispatch_io_create_with_path(DISPATCH_IO_RANDOM,
        [path fileSystemRepresentation], O_RDONLY, 0, queue , NULL);
    dispatch_io_set_low_water(io, SIZE_MAX);
    dispatch_io_read(io, 0, SIZE_MAX, queue,
        ^(bool done, dispatch_data_t data, int error){
            if (error == 0){
                NSImage *image = [[NSImage alloc] initWithData:(NSData*)data];
                @synchronized(images){ [images addObject:image]; }
            }
        });
}
```

# Dispatch IO

## Reading many files

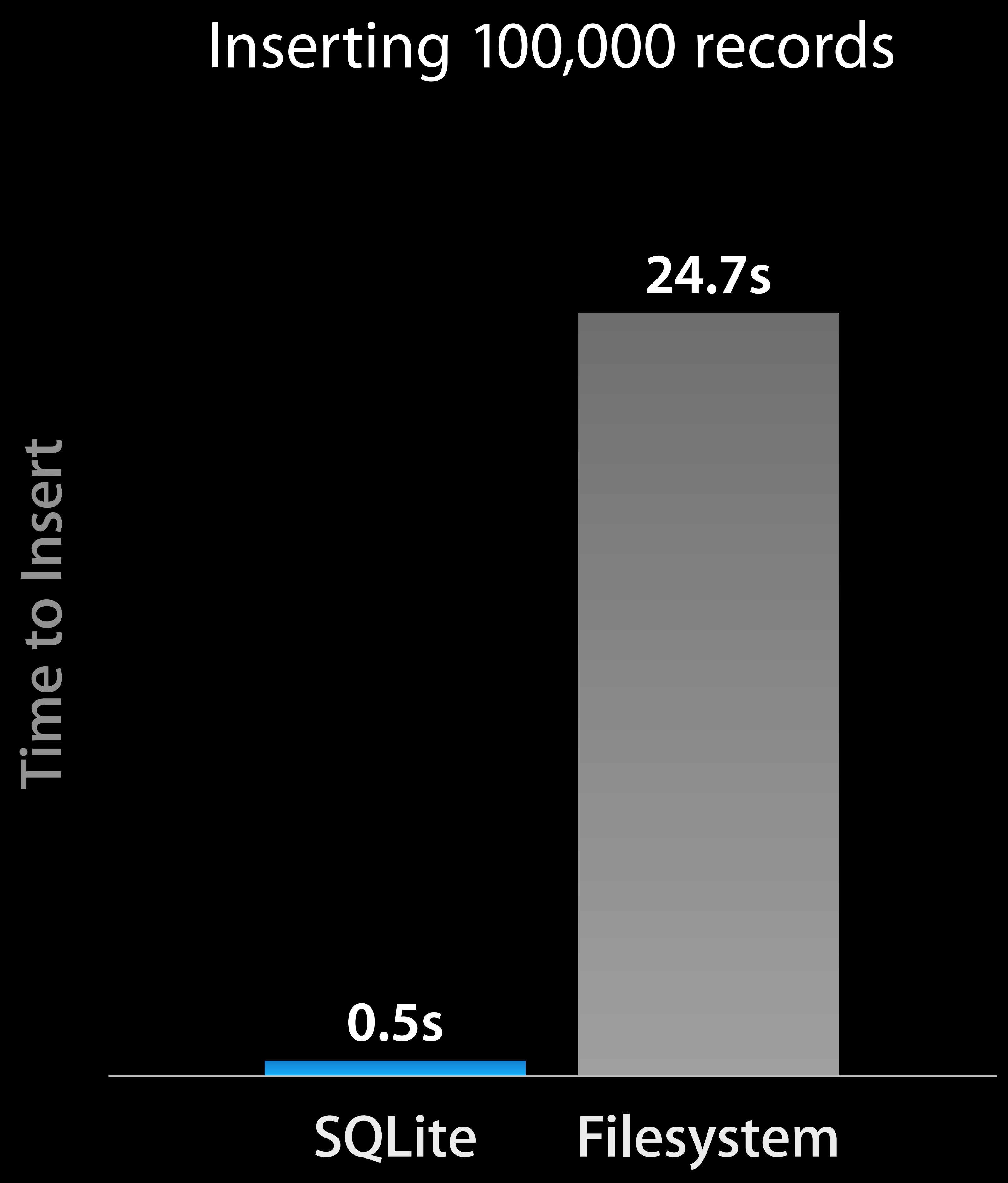
```
dispatch_queue_t queue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_LOW,0);
for (NSString *path in imagePaths) {
    dispatch_io_t io = dispatch_io_create_with_path(DISPATCH_IO_RANDOM,
        [path fileSystemRepresentation], O_RDONLY, 0, queue , NULL);
    dispatch_io_set_low_water(io, SIZE_MAX);
    dispatch_io_read(io, 0, SIZE_MAX, queue,
        ^(bool done, dispatch_data_t data, int error){
            if (error == 0){
                NSImage *image = [[NSImage alloc] initWithData:(NSData*)data];
                @synchronized(images){ [images addObject:image]; }
            }
        });
}
```

# Organizing Data on Disk

- Storing large numbers of small files is expensive
- Use Core Data or sqlite to store small objects
  - Control over atomicity
  - More space efficient
  - Better query capabilities

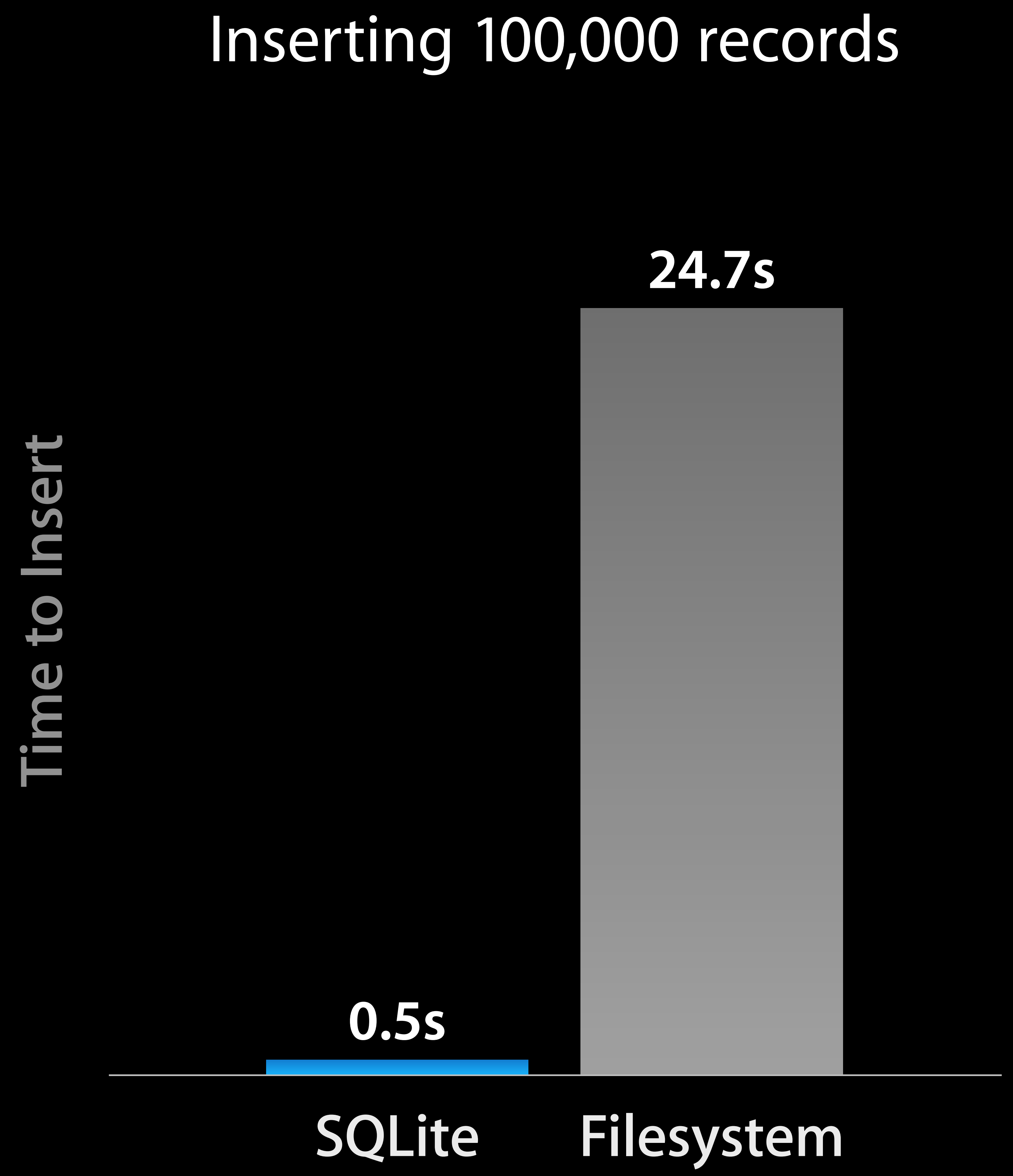
# Organizing Data on Disk

- Storing large numbers of small files is expensive
- Use Core Data or sqlite to store small objects
  - Control over atomicity
  - More space efficient
  - Better query capabilities



# Organizing Data on Disk

- Storing large numbers of small files is expensive
- Use Core Data or sqlite to store small objects
  - Control over atomicity
  - More space efficient
  - Better query capabilities



# Write Buffering

```
int fd = open("/tmp/foo", O_CREAT | O_WRONLY, 0755);
write(fd, buf, FILE_SIZE);
close(fd);
```

- Use CoreData/sqlite if you need consistency guarantees

# Write Buffering

```
int fd = open("/tmp/foo", O_CREAT | O_WRONLY, 0755);
write(fd, buf, FILE_SIZE);
close(fd); // write is issued here
```

- Use CoreData/sqlite if you need consistency guarantees

# Write Buffering

```
int fd = open("/tmp/foo", O_CREAT | O_WRONLY, 0755);
write(fd, buf, FILE_SIZE);
close(fd); // write is issued here
```

| | |
|---|---|
| VFS | `close()`<br>`fsync()` |
| Memory Mapped IO | `msync()` |

- Use CoreData/sqlite if you need consistency guarantees

# Write Buffering

```
int fd = open("/tmp/foo", O_CREAT | O_WRONLY, 0755);
write(fd, buf, FILE_SIZE);
close(fd); // write is issued here
```

| | |
|---|---|
| VFS | `close()`<br>`fsync()` |
| Memory Mapped IO | `msync()` |

- Use CoreData/sqlite if you need consistency guarantees

# File Cache Management

- Cached IO is >100x faster

# File Cache Management

- Cached IO is >100x faster
- File cache competes for memory
- Use non-cached IO when data won't be needed again
    - e.g. reading an archive to extract it, streaming large multimedia files

```
[NSData dataWithContentsOfFile: p
options: NSDataReadingUncached error:&e]

fcntl(fd, F_NOCACHE, 1);
// file descriptor can then be passed to dispatch_io_create
```

# Memory Mapped IO

- Avoid another copy of data
- Ideal for random reads
- madvise() can be used to indicate future data needs

```
[NSData dataWithContentsOfURL: aURL
    options: NSDataReadingMappedIfSafe error:&error]

mmap(NULL, size, PROT_READ, MAP_SHARED, fd, 0);
```

Don’t do IO on the main thread!

# Profiling Disk Access

## fs_usage

- `fs_usage [-w] [-f mode] [-t seconds] [pid | cmd]`
  - Filter by type of events with -f <mode>
    - filesys – all filesystem events
    - diskio – IOs that access disks
  - Use -w to force wide output when redirecting to a file

# Profiling Disk Access

## Decoding fs_usage

```
$ sudo fs_usage -f filesys
```

```
02:53:00.640031  open            F=3        (R_____)  5/36b460f00575b2308f849f2981bb5ad  0.000005 git.827453
02:53:00.640032  fstat64         F=3                                                     0.000001 git.827453
02:53:00.640035  mmap            F=3    A=0x0122bc3000  O=0x00000000   B=0x1000 <READ>   0.000003 git.827453
02:53:00.640036  close           F=3                                                     0.000002 git.827453
02:53:02.236841  pread           F=40   B=0x20       O=0x00000180                        0.000002 Safari.827472
02:53:02.236843  pread           F=40   B=0x40       O=0x000000c0                        0.000001 Safari.827472
02:53:02.236858  pread           F=40   B=0x400      O=0x001ab800                        0.000002 Safari.827472
02:53:02.238335    RdData[A]     D=0x05ad6150  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.001454 Safari.827472
02:53:02.238359  pread           F=40   B=0x20       O=0x0003cd00                        0.001495 Safari.827472
02:53:02.238795    RdData[A]     D=0x07bf6888  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.000395 Safari.827472
02:53:02.240151  pread           F=40   B=0x1000     O=0x005ee000                        0.000008 Safari.827472
02:53:02.569863    RdData[AN]    D=0x16e35980  B=0x11000  /dev/disk1                     0.001703 iTunes.824697
02:53:02.569905  pread           F=40   B=0x10000    O=0x00bf9060                        0.001780 iTunes.824697
```

# Profiling Disk Access

## Decoding fs_usage

```
$ sudo fs_usage -f filesys
```

```
02:53:00.640031  open            F=3        (R_____)  5/36b460f00575b2308f849f2981bb5ad  0.000005 git.827453
02:53:00.640032  fstat64         F=3                                                     0.000001 git.827453
02:53:00.640035  mmap            F=3    A=0x0122bc3000  O=0x00000000   B=0x1000 <READ>   0.000003 git.827453
02:53:00.640036  close           F=3                                                     0.000002 git.827453
02:53:02.236841  pread           F=40   B=0x20       O=0x00000180                        0.000002 Safari.827472
02:53:02.236843  pread           F=40   B=0x40       O=0x000000c0                        0.000001 Safari.827472
02:53:02.236858  pread           F=40   B=0x400      O=0x001ab800                        0.000002 Safari.827472
02:53:02.238335    RdData[A]     D=0x05ad6150  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.001454 Safari.827472
02:53:02.238359  pread           F=40   B=0x20       O=0x0003cd00                        0.001495 Safari.827472
02:53:02.238795    RdData[A]     D=0x07bf6888  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.000395 Safari.827472
02:53:02.240151  pread           F=40   B=0x1000     O=0x005ee000                        0.000008 Safari.827472
02:53:02.569863    RdData[AN]    D=0x16e35980  B=0x11000  /dev/disk1                     0.001703 iTunes.824697
02:53:02.569905  pread           F=40   B=0x10000    O=0x00bf9060                        0.001780 iTunes.824697
```

- Completion Time

# Profiling Disk Access

## Decoding fs_usage

```
$ sudo fs_usage -f filesys
```

```
02:53:00.640031  open            F=3        (R_____)  5/36b460f00575b2308f849f2981bb5ad  0.000005 git.827453
02:53:00.640032  fstat64         F=3                                                     0.000001 git.827453
02:53:00.640035  mmap            F=3    A=0x0122bc3000  O=0x00000000   B=0x1000 <READ>   0.000003 git.827453
02:53:00.640036  close           F=3                                                     0.000002 git.827453
02:53:02.236841  pread           F=40   B=0x20       O=0x00000180                        0.000002 Safari.827472
02:53:02.236843  pread           F=40   B=0x40       O=0x000000c0                        0.000001 Safari.827472
02:53:02.236858  pread           F=40   B=0x400      O=0x001ab800                        0.000002 Safari.827472
02:53:02.238335    RdData[A]     D=0x05ad6150  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.001454 Safari.827472
02:53:02.238359  pread           F=40   B=0x20       O=0x0003cd00                        0.001495 Safari.827472
02:53:02.238795    RdData[A]     D=0x07bf6888  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.000395 Safari.827472
02:53:02.240151  pread           F=40   B=0x1000     O=0x005ee000                        0.000008 Safari.827472
02:53:02.569863    RdData[AN]    D=0x16e35980  B=0x11000  /dev/disk1                     0.001703 iTunes.824697
02:53:02.569905  pread           F=40   B=0x10000    O=0x00bf9060                        0.001780 iTunes.824697
```

- Completion Time
- System Call / Event

# Profiling Disk Access

## Decoding fs_usage

```
$ sudo fs_usage -f filesys
```

```
02:53:00.640031  open            F=3        (R_____)  5/36b460f00575b2308f849f2981bb5ad  0.000005 git.827453
02:53:00.640032  fstat64         F=3                                                     0.000001 git.827453
02:53:00.640035  mmap            F=3    A=0x0122bc3000  O=0x00000000   B=0x1000 <READ>   0.000003 git.827453
02:53:00.640036  close           F=3                                                     0.000002 git.827453
02:53:02.236841  pread           F=40   B=0x20       O=0x00000180                        0.000002 Safari.827472
02:53:02.236843  pread           F=40   B=0x40       O=0x000000c0                        0.000001 Safari.827472
02:53:02.236858  pread           F=40   B=0x400      O=0x001ab800                        0.000002 Safari.827472
02:53:02.238335    RdData[A]     D=0x05ad6150  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.001454 Safari.827472
02:53:02.238359  pread           F=40   B=0x20       O=0x0003cd00                        0.001495 Safari.827472
02:53:02.238795    RdData[A]     D=0x07bf6888  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.000395 Safari.827472
02:53:02.240151  pread           F=40   B=0x1000     O=0x005ee000                        0.000008 Safari.827472
02:53:02.569863    RdData[AN]    D=0x16e35980  B=0x11000  /dev/disk1                     0.001703 iTunes.824697
02:53:02.569905  pread           F=40   B=0x10000    O=0x00bf9060                        0.001780 iTunes.824697
```

- Completion Time
- System Call / Event
- Event Details

# Profiling Disk Access

## Decoding fs_usage

```
$ sudo fs_usage -f filesys
```

```
02:53:00.640031  open            F=3        (R_____)  5/36b460f00575b2308f849f2981bb5ad  0.000005 git.827453
02:53:00.640032  fstat64         F=3                                                     0.000001 git.827453
02:53:00.640035  mmap            F=3    A=0x0122bc3000  O=0x00000000   B=0x1000 <READ>   0.000003 git.827453
02:53:00.640036  close           F=3                                                     0.000002 git.827453
02:53:02.236841  pread           F=40   B=0x20       O=0x00000180                        0.000002 Safari.827472
02:53:02.236843  pread           F=40   B=0x40       O=0x000000c0                        0.000001 Safari.827472
02:53:02.236858  pread           F=40   B=0x400      O=0x001ab800                        0.000002 Safari.827472
02:53:02.238335    RdData[A]     D=0x05ad6150  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.001454 Safari.827472
02:53:02.238359  pread           F=40   B=0x20       O=0x0003cd00                        0.001495 Safari.827472
02:53:02.238795    RdData[A]     D=0x07bf6888  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.000395 Safari.827472
02:53:02.240151  pread           F=40   B=0x1000     O=0x005ee000                        0.000008 Safari.827472
02:53:02.569863    RdData[AN]    D=0x16e35980  B=0x11000  /dev/disk1                     0.001703 iTunes.824697
02:53:02.569905  pread           F=40   B=0x10000    O=0x00bf9060                        0.001780 iTunes.824697
```

- Completion Time
- System Call / Event
- Event Details
- Duration

# Profiling Disk Access

## Decoding fs_usage

```
$ sudo fs_usage -f filesys
```

```
02:53:00.640031  open            F=3        (R_____)  5/36b460f00575b2308f849f2981bb5ad  0.000005 git.827453
02:53:00.640032  fstat64         F=3                                                     0.000001 git.827453
02:53:00.640035  mmap            F=3    A=0x0122bc3000  O=0x00000000   B=0x1000 <READ>   0.000003 git.827453
02:53:00.640036  close           F=3                                                     0.000002 git.827453
02:53:02.236841  pread           F=40   B=0x20       O=0x00000180                        0.000002 Safari.827472
02:53:02.236843  pread           F=40   B=0x40       O=0x000000c0                        0.000001 Safari.827472
02:53:02.236858  pread           F=40   B=0x400      O=0x001ab800                        0.000002 Safari.827472
02:53:02.238335    RdData[A]     D=0x05ad6150  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.001454 Safari.827472
02:53:02.238359  pread           F=40   B=0x20       O=0x0003cd00                        0.001495 Safari.827472
02:53:02.238795    RdData[A]     D=0x07bf6888  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.000395 Safari.827472
02:53:02.240151  pread           F=40   B=0x1000     O=0x005ee000                        0.000008 Safari.827472
02:53:02.569863    RdData[AN]    D=0x16e35980  B=0x11000  /dev/disk1                     0.001703 iTunes.824697
02:53:02.569905  pread           F=40   B=0x10000    O=0x00bf9060                        0.001780 iTunes.824697
```

- Completion Time
- System Call / Event
- Event Details
- Duration
- Process and Thread ID

# Profiling Disk Access

## Decoding fs_usage

```
$ sudo fs_usage -f filesys
```

```
02:53:00.640031  open            F=3        (R_____)  5/36b460f00575b2308f849f2981bb5ad  0.000005 git.827453
02:53:00.640032  fstat64         F=3                                                     0.000001 git.827453
02:53:00.640035  mmap            F=3    A=0x0122bc3000  O=0x00000000   B=0x1000 <READ>   0.000003 git.827453
02:53:00.640036  close           F=3                                                     0.000002 git.827453
02:53:02.236841  pread           F=40   B=0x20       O=0x00000180                        0.000002 Safari.827472
02:53:02.236843  pread           F=40   B=0x40       O=0x000000c0                        0.000001 Safari.827472
02:53:02.236858  pread           F=40   B=0x400      O=0x001ab800                        0.000002 Safari.827472
02:53:02.238335    RdData[A]     D=0x05ad6150  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.001454 Safari.827472
02:53:02.238359  pread           F=40   B=0x20       O=0x0003cd00                        0.001495 Safari.827472
02:53:02.238795    RdData[A]     D=0x07bf6888  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.000395 Safari.827472
02:53:02.240151  pread           F=40   B=0x1000     O=0x005ee000                        0.000008 Safari.827472
02:53:02.569863    RdData[AN]    D=0x16e35980  B=0x11000  /dev/disk1                     0.001703 iTunes.824697
02:53:02.569905  pread           F=40   B=0x10000    O=0x00bf9060                        0.001780 iTunes.824697
```

- Completion Time
- System Call / Event
- Event Details
- Duration
- Process and Thread ID

# Profiling Disk Access

## Decoding fs_usage

```
$ sudo fs_usage -f filesys
```

```
02:53:00.640031  open            F=3        (R_____)  5/36b460f00575b2308f849f2981bb5ad  0.000005 git.827453
02:53:00.640032  fstat64         F=3                                                     0.000001 git.827453
02:53:00.640035  mmap            F=3    A=0x0122bc3000  O=0x00000000   B=0x1000 <READ>   0.000003 git.827453
02:53:00.640036  close           F=3                                                     0.000002 git.827453
02:53:02.236841  pread           F=40   B=0x20       O=0x00000180                        0.000002 Safari.827472
02:53:02.236843  pread           F=40   B=0x40       O=0x000000c0                        0.000001 Safari.827472
02:53:02.236858  pread           F=40   B=0x400      O=0x001ab800                        0.000002 Safari.827472
02:53:02.238335    RdData[A]     D=0x05ad6150  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.001454 Safari.827472
02:53:02.238359  pread           F=40   B=0x20       O=0x0003cd00                        0.001495 Safari.827472
02:53:02.238795    RdData[A]     D=0x07bf6888  B=0x1000   /dev/disk1  y/Safari/HistoryIn 0.000395 Safari.827472
02:53:02.240151  pread           F=40   B=0x1000     O=0x005ee000                        0.000008 Safari.827472
02:53:02.569863    RdData[AN]    D=0x16e35980  B=0x11000  /dev/disk1                     0.001703 iTunes.824697
02:53:02.569905  pread           F=40   B=0x10000    O=0x00bf9060                        0.001780 iTunes.824697
```

- Completion Time
- System Call / Event
- Event Details
- Duration
- Process and Thread ID

# Profiling Disk Access

## diskio lines

```
$ sudo fs_usage −f diskio
```

```
15:38:19.677656  WrMeta[AT3]  D=0x00509740  B=0x2000    /dev/disk1  /private/var/log/              0.000286  launchd.284
15:38:20.281154  RdData[AN]   D=0x0e5bfaa0  B=0x11000   /dev/disk1                                 0.001635  iTunes.585253
15:38:20.574564  RdData[AP]   D=0x09f36bc0  B=0x1000    /dev/disk1  private/var/log/powermanageme  0.000364  syslogd.587090
15:38:22.022556  RdData[AN]   D=0x0e5bfb20  B=0x11000   /dev/disk1                                 0.001597  iTunes.585253
15:38:22.120809  WrData[AP]   D=0x05c697f8  B=0x1000    /dev/disk1  private/var/log/powermanageme  0.000166  Keynote.587358
15:38:23.690691  RdData[A]    D=0x0569bc48  B=0x1000    /dev/disk1                                 0.000284  Mail.587502
```

# Profiling Disk Access

## diskio lines

```
$ sudo fs_usage -f diskio
15:38:19.677656  WrMeta[AT3]  D=0x00509740  B=0x2000    /dev/disk1  /private/var/log/              0.000286  launchd.284
15:38:20.281154  RdData[AN]   D=0x0e5bfaa0  B=0x11000   /dev/disk1                                 0.001635  iTunes.585253
15:38:20.574564  RdData[AP]   D=0x09f36bc0  B=0x1000    /dev/disk1  private/var/log/powermanageme  0.000364  syslogd.587090
15:38:22.022556  RdData[AN]   D=0x0e5bfb20  B=0x11000   /dev/disk1                                 0.001597  iTunes.585253
15:38:22.120809  WrData[AP]   D=0x05c697f8  B=0x1000    /dev/disk1  private/var/log/powermanageme  0.000166  Keynote.587358
15:38:23.690691  RdData[A]    D=0x0569bc48  B=0x1000    /dev/disk1                                 0.000284  Mail.587502
```

- Type of IO:
    - Wr = Write, Rd = Read
    - Data = File Data, Meta = Filesystem Metadata
    - PgIn = Read from file-backed memory, PgOut = Write
    - N = non-cached

# Profiling Disk Access

## diskio lines

```
$ sudo fs_usage -f diskio
```

```
15:38:19.677656  WrMeta[AT3]  D=0x00509740  B=0x2000    /dev/disk1  /private/var/log/            0.000286  launchd.284
15:38:20.281154  RdData[AN]   D=0x0e5bfaa0  B=0x11000   /dev/disk1                               0.001635  iTunes.585253
15:38:20.574564  RdData[AP]   D=0x09f36bc0  B=0x1000    /dev/disk1  private/var/log/powermanageme  0.000364  syslogd.587090
15:38:22.022556  RdData[AN]   D=0x0e5bfb20  B=0x11000   /dev/disk1                               0.001597  iTunes.585253
15:38:22.120809  WrData[AP]   D=0x05c697f8  B=0x1000    /dev/disk1  private/var/log/powermanageme  0.000166  Keynote.587358
15:38:23.690691  RdData[A]    D=0x0569bc48  B=0x1000    /dev/disk1                               0.000284  Mail.587502
```

- Type of IO:
  - Wr = Write, Rd = Read
  - Data = File Data, Meta = Filesystem Metadata
  - PgIn = Read from file-backed memory, PgOut = Write
  - N = non-cached
- D=offset on disk

# Profiling Disk Access

## diskio lines

```
$ sudo fs_usage -f diskio
```

```
15:38:19.677656  WrMeta[AT3]   D=0x00509740  B=0x2000     /dev/disk1  /private/var/log/            0.000286  launchd.284
15:38:20.281154  RdData[AN]    D=0x0e5bfaa0  B=0x11000    /dev/disk1                               0.001635  iTunes.585253
15:38:20.574564  RdData[AP]    D=0x09f36bc0  B=0x1000     /dev/disk1  private/var/log/powermanageme  0.000364  syslogd.587090
15:38:22.022556  RdData[AN]    D=0x0e5bfb20  B=0x11000    /dev/disk1                               0.001597  iTunes.585253
15:38:22.120809  WrData[AP]    D=0x05c697f8  B=0x1000     /dev/disk1  private/var/log/powermanageme  0.000166  Keynote.587358
15:38:23.690691  RdData[A]     D=0x0569bc48  B=0x1000     /dev/disk1                               0.000284  Mail.587502
```

- Type of IO:
  - Wr = Write, Rd = Read
  - Data = File Data, Meta = Filesystem Metadata
  - PgIn = Read from file-backed memory, PgOut = Write
  - N = non-cached
- D=offset on disk
- B=size

# Profiling Disk Access

## diskio lines

```
$ sudo fs_usage -f diskio
```

```
15:38:19.677656  WrMeta[AT3]  D=0x00509740  B=0x2000    /dev/disk1  /private/var/log/              0.000286  launchd.284
15:38:20.281154  RdData[AN]   D=0x0e5bfaa0  B=0x11000   /dev/disk1                                 0.001635  iTunes.585253
15:38:20.574564  RdData[AP]   D=0x09f36bc0  B=0x1000    /dev/disk1  private/var/log/powermanageme  0.000364  syslogd.587090
15:38:22.022556  RdData[AN]   D=0x0e5bfb20  B=0x11000   /dev/disk1                                 0.001597  iTunes.585253
15:38:22.120809  WrData[AP]   D=0x05c697f8  B=0x1000    /dev/disk1  private/var/log/powermanageme  0.000166  Keynote.587358
15:38:23.690691  RdData[A]    D=0x0569bc48  B=0x1000    /dev/disk1                                 0.000284  Mail.587502
```

- Type of IO:
  - Wr = Write, Rd = Read
  - Data = File Data, Meta = Filesystem Metadata
  - PgIn = Read from file-backed memory, PgOut = Write
  - N = non-cached
- D=offset on disk
- B=size
- Disk

# Profiling Disk Access

## diskio lines

```
$ sudo fs_usage -f diskio
```

```
15:38:19.677656  WrMeta[AT3]  D=0x00509740  B=0x2000    /dev/disk1  /private/var/log/             0.000286  launchd.284
15:38:20.281154  RdData[AN]   D=0x0e5bfaa0  B=0x11000   /dev/disk1                                0.001635  iTunes.585253
15:38:20.574564  RdData[AP]   D=0x09f36bc0  B=0x1000    /dev/disk1  private/var/log/powermanageme  0.000364  syslogd.587090
15:38:22.022556  RdData[AN]   D=0x0e5bfb20  B=0x11000   /dev/disk1                                0.001597  iTunes.585253
15:38:22.120809  WrData[AP]   D=0x05c697f8  B=0x1000    /dev/disk1  private/var/log/powermanageme  0.000166  Keynote.587358
15:38:23.690691  RdData[A]    D=0x0569bc48  B=0x1000    /dev/disk1                                0.000284  Mail.587502
```

- Type of IO:
  - Wr = Write, Rd = Read
  - Data = File Data, Meta = Filesystem Metadata
  - PgIn = Read from file-backed memory, PgOut = Write
  - N = non-cached
- D=offset on disk
- B=size
- Disk
- Filename, if available

# Profiling Disk Access

## Improving Performance

# Profiling Disk Access

## Improving Performance

- Don’t do it

# Profiling Disk Access

## Improving Performance

- Don’t do it
- Do it less

# Profiling Disk Access

## Improving Performance

- Don’t do it
- Do it less
- Do it later

# Profiling Disk Access

## Improving Performance

- Don’t do it
- Do it less
- Do it later
- Do it sequentially

# Impact of the Disk Cache

# Impact of the Disk Cache

## Warm App Launch

```
21:52:46.595005    RdData[AP]      D=0x0dd68050  B=0x1000   /dev/disk2  d Application State/com.apple.Console.savedState/windows.plist    0.000524 W Console.51388
21:52:46.647442    WrData[AP]      D=0x0dd7c980  B=0x1000   /dev/disk2  lication State/com.apple.Console.savedState/restorecount.plist    0.000356 W Console.51385
21:52:46.801626    WrData[AP]      D=0x0dd7c980  B=0x1000   /dev/disk2  lication State/com.apple.Console.savedState/restorecount.plist    0.000394 W Console.51391
21:52:48.513875    WrData[AP]      D=0x0dd7c990  B=0x1000   /dev/disk2  Saved Application State/com.apple.Console.savedState/data.data    0.001438 W Console.51397
21:52:48.513884    WrData[ANP]     D=0x0dd7c998  B=0x1000   /dev/disk2  d Application State/com.apple.Console.savedState/windows.plist    0.001263 W Console.51397
21:52:48.516574    WrData[ANP]     D=0x0dd7c9a0  B=0x3000   /dev/disk2  d Application State/com.apple.Console.savedState/window_1.data    0.000115 W Console.51397
21:52:48.720244    WrData[ANP]     D=0x0dd7c9b8  B=0xc0000  /dev/disk2  d Application State/com.apple.Console.savedState/window_2.data    0.003129 W Console.51388
```

# Impact of the Disk Cache

## Warm App Launch

```
21:52:46.595005    RdData[AP]      D=0x0dd68050  B=0x1000   /dev/disk2  d Application State/com.apple.Console.savedState/windows.plist    0.000524 W Console.51388
21:52:46.647442    WrData[AP]      D=0x0dd7c980  B=0x1000   /dev/disk2  lication State/com.apple.Console.savedState/restorecount.plist    0.000356 W Console.51385
21:52:46.801626    WrData[AP]      D=0x0dd7c980  B=0x1000   /dev/disk2  lication State/com.apple.Console.savedState/restorecount.plist    0.000394 W Console.51391
21:52:48.513875    WrData[AP]      D=0x0dd7c990  B=0x1000   /dev/disk2  Saved Application State/com.apple.Console.savedState/data.data    0.001438 W Console.51397
21:52:48.513884    WrData[ANP]     D=0x0dd7c998  B=0x1000   /dev/disk2  d Application State/com.apple.Console.savedState/windows.plist    0.001263 W Console.51397
21:52:48.516574    WrData[ANP]     D=0x0dd7c9a0  B=0x3000   /dev/disk2  d Application State/com.apple.Console.savedState/window_1.data    0.000115 W Console.51397
21:52:48.720244    WrData[ANP]     D=0x0dd7c9b8  B=0xc0000  /dev/disk2  d Application State/com.apple.Console.savedState/window_2.data    0.003129 W Console.51388
```

## Cold App Launch

```
21:50:35.157462    RdData[A]       D=0x0bf00020  B=0x1000   /dev/disk2  ar/db/launchd.db/com.apple.launchd.peruser.502/overrides.plist    0.000272 W open.50607
21:50:35.160401    RdMeta[ST1]     D=0x001aade0  B=0x2000   /dev/disk2                                                                    0.000267 W launchd.50616
21:50:35.166417    RdMeta[ST1]     D=0x001aadd0  B=0x2000   /dev/disk2                                                                    0.000367 W launchd.50616
21:50:35.172389    RdMeta[ST1]     D=0x02c52fa8  B=0x2000   /dev/disk2                                                                    0.000296 W launchd.50616
21:50:35.172768    RdMeta[ST1]     D=0x00254a30  B=0x2000   /dev/disk2                                                                    0.000294 W launchd.50616
21:50:35.173033    RdData[AT1]     D=0x0ab92768  B=0x1000   /dev/disk2                                                                    0.000212 W launchd.50616
21:50:35.173430    RdData[AT1]     D=0x0ab92770  B=0x7000   /dev/disk2  /Utilities/Console.app/Contents/MacOS/Console/..namedfork/rsrc    0.000330 W launchd.50616
21:50:35.174239    RdData[AT1]     D=0x0ab92880  B=0x5000   /dev/disk2  /Utilities/Console.app/Contents/MacOS/Console/..namedfork/rsrc    0.000328 W launchd.50616
21:50:35.174676    RdData[AT1]     D=0x0ab928a8  B=0x1000   /dev/disk2  /Utilities/Console.app/Contents/MacOS/Console/..namedfork/rsrc    0.000231 W launchd.50616
21:50:35.177371    RdMeta[S]       D=0x001aacc0  B=0x2000   /dev/disk2                                                                    0.000305 W Dock.5011
21:50:35.177933    RdMeta[S]       D=0x02c67e68  B=0x2000   /dev/disk2                                                                    0.000264 W Console.50616
21:50:35.177966    RdMeta[S]       D=0x001aadf0  B=0x2000   /dev/disk2                                                                    0.000281 W Dock.5011
21:50:35.178398    RdMeta[S]       D=0x0006fca0  B=0x2000   /dev/disk2                                                                    0.000242 W Dock.5011
21:50:35.178673    RdMeta[S]       D=0x0006f700  B=0x2000   /dev/disk2                                                                    0.000258 W Dock.5011
21:50:35.179110    RdData[A]       D=0x0ab92858  B=0x5000   /dev/disk2  /Utilities/Console.app/Contents/MacOS/Console/..namedfork/rsrc    0.000308 W Console.50616
21:50:35.179317    RdMeta[S]       D=0x0006f6f0  B=0x2000   /dev/disk2                                                                    0.000296 W Dock.5011
21:50:35.183878    WrData[A]       D=0x0ee00468  B=0x40000  /dev/disk2  apple.IconServices/D74617D79809E180C33093851CCD3FC6.iscachebmp    0.000677 W com.apple.IconS.50630
21:50:35.185039    RdData[A]       D=0x0ab92820  B=0x7000   /dev/disk2  /Utilities/Console.app/Contents/MacOS/Console/..namedfork/rsrc    0.000338 W Console.50616
21:50:35.192602    PgIn[A]         D=0x06334fc0  B=0x1000   /dev/disk2  /Users/anthony/Library/Preferences/com.apple.Console.plist        0.000267 W cfprefsd.50602
21:50:35.209777    RdMeta[S]       D=0x001a2970  B=0x2000   /dev/disk2                                                                    0.000333 W Console.50616
21:50:35.210039    RdMeta[S]       D=0x0b1e3788  B=0x1000   /dev/disk2                                                                    0.000224 W Console.50616
21:50:35.210295    RdMeta[S]       D=0x0b1e37b0  B=0x1000   /dev/disk2                                                                    0.000224 W Console.50616
21:50:35.212187    RdData[A]       D=0x0b5f29e0  B=0x1000   /dev/disk2                                                                    0.000233 W Console.50616
21:50:35.212508    RdData[A]       D=0x0b5f29c8  B=0x1000   /dev/disk2                                                                    0.000221 W Console.50616
21:50:35.212799    RdData[A]       D=0x0b5f29d0  B=0x2000   /dev/disk2                                                                    0.000230 W Console.50616
21:50:35.219963    RdData[A]       D=0x0ab927a8  B=0x8000   /dev/disk2  /Utilities/Console.app/Contents/MacOS/Console/..namedfork/rsrc    0.000451 W Console.50616
21:50:35.220697    RdData[A]       D=0x0ab927e8  B=0x7000   /dev/disk2  /Utilities/Console.app/Contents/MacOS/Console/..namedfork/rsrc    0.000336 W Console.50616
21:50:35.223572    PgIn[A]         D=0x0cf22068  B=0x4000   /dev/disk2                                                                    0.000275 W Console.50616
21:50:35.224143    PgIn[A]         D=0x0cf219b0  B=0x10000  /dev/disk2                                                                    0.000472 W Console.50616
21:50:35.224692    RdMeta[S]       D=0x002624c0  B=0x2000   /dev/disk2                                                                    0.000281 W Console.50616
```

# Impact of the Disk Cache

# Impact of the Disk Cache

- Profile in different warmth states
- Use the purge command to evict caches
- Some data may be pre-warmed at boot

# Disk IO Recap

- Use dispatch IO
- Profile your disk access in different warmth states
- Use non-cached IO when accessing data only once
- Pay attention to when data is flushed
- Don’t do IO on the main thread

# Working in the Background

# Background Work

- Apps do background work
  - Refreshing or syncing user data
  - Indexing or backing up a user's files
- This hurts system responsiveness
- Backgrounding limits resource use

# Backgrounding Effects

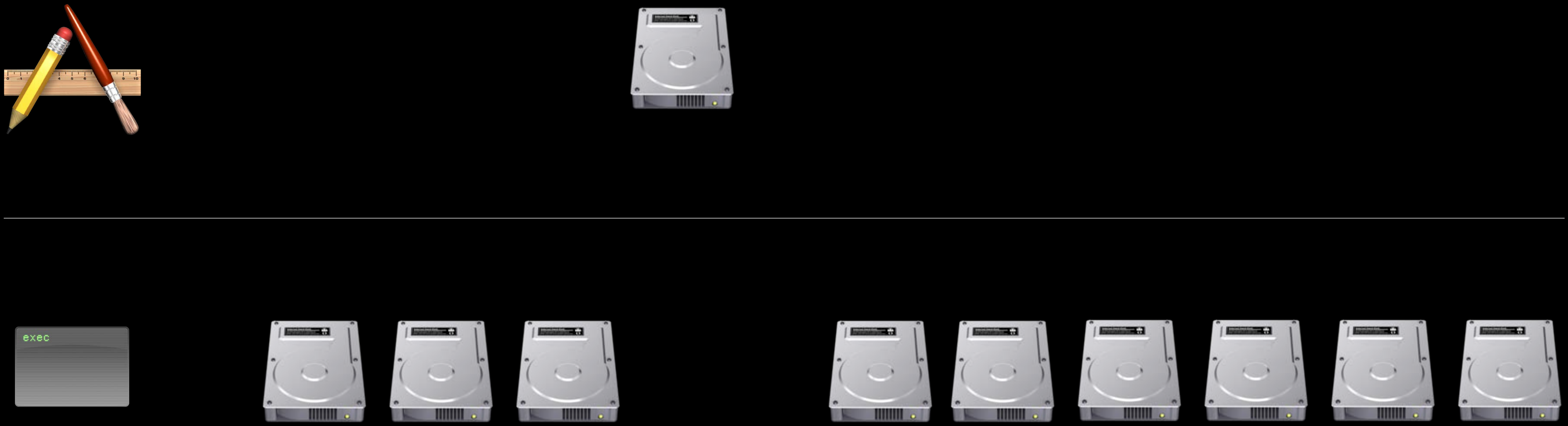
- Hints to perform work more efficiently
- Lowered CPU scheduling priority
- IO Throttling

# Backgrounding Effects

- Hints to perform work more efficiently
- Lowered CPU scheduling priority
- IO Throttling

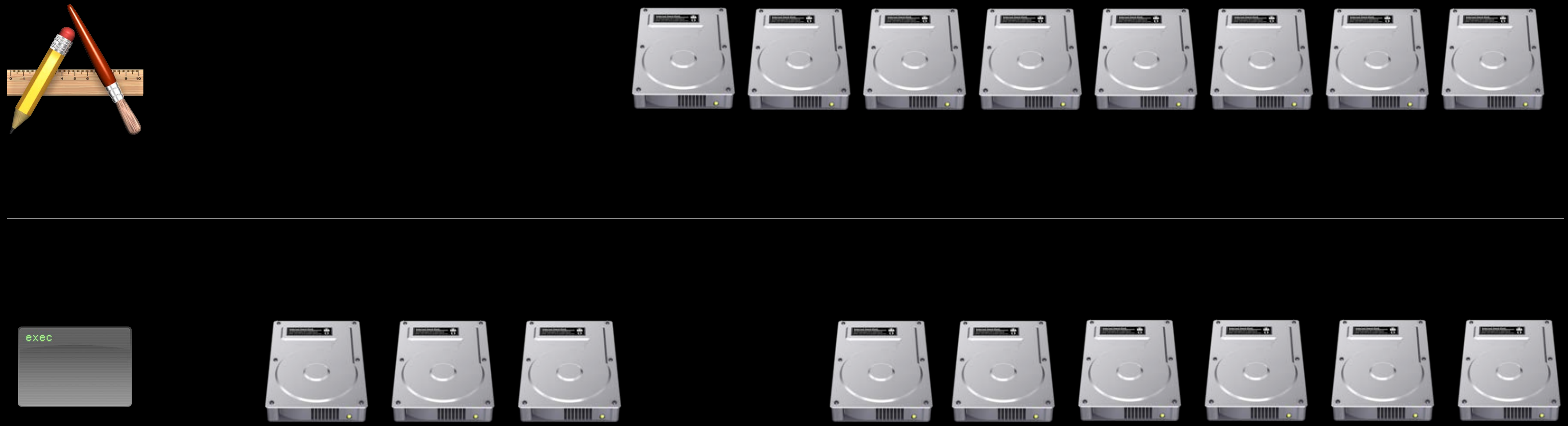
# Backgrounding Effects

- Hints to perform work more efficiently
- Lowered CPU scheduling priority
- IO Throttling



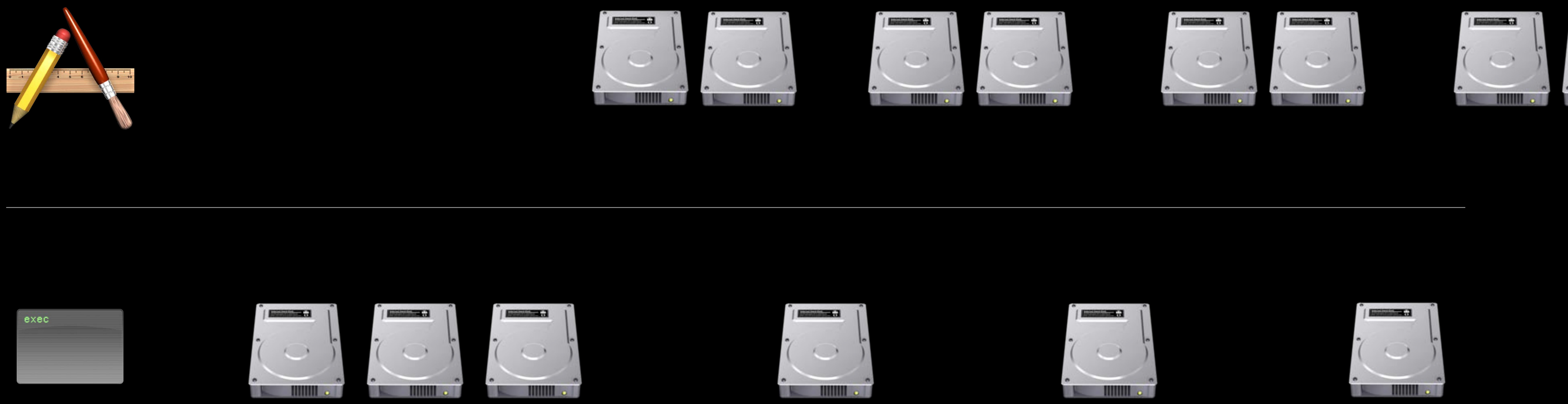
# Backgrounding Effects

- Hints to perform work more efficiently
- Lowered CPU scheduling priority
- IO Throttling



# Backgrounding Effects

- Hints to perform work more efficiently
- Lowered CPU scheduling priority
- IO Throttling



# Backgrounding a Block

## Background priority dispatch queue

```
dispatch_queue_t bgQueue =
dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_BACKGROUND, 0);

dispatch_async(bgQueue, ^{
    /* XXX: this code should not take locks needed by UI */

    /* your expensive, background work here */

});
```

# Backgrounding Large Tasks

## Use XPC

# Backgrounding Large Tasks

## Use XPC

- XPC Activity
  - Let the system pick the best time to perform a task

# Backgrounding Large Tasks

## Use XPC

- XPC Activity
  - Let the system pick the best time to perform a task
- Adaptive Daemon
  - XPC Services run in background by default
  - Boosted out of background upon app's message

Efficient Design with XPC

Russian Hill
Tuesday 2:00PM

# Background Continuous Work

## Thread/Process adoption

- Use launchd’s Background ProcessType

```
<key>ProcessType</key>
<string>Background</string>
```

- Use setpriority(3)

```
setpriority(PRIO_DARWIN_PROCESS, 0, PRIO_DARWIN_BG);
```

# Debugging Backgrounding

- ps -aMx will show priority – background is 4 or less

```
anthony     1547  ??  0.0 S   4T   0:00.01  0:00.02 <process name>
            1547      0.0 S   4T   0:00.00  0:00.00
            1547      0.0 S   4T   0:00.00  0:00.00
```

# Debugging Backgrounding

- ps -aMx will show priority – background is 4 or less

```
anthony     1547  ??  0.0 S   4T   0:00.01  0:00.02 <process name>
            1547      0.0 S   4T   0:00.00  0:00.00
            1547      0.0 S   4T   0:00.00  0:00.00
```

- spindump – look for throttle_lowpri_io frame

```
Process:         accountsd [242]
Importance:      Adaptive, Background Priority
```

# Debugging Backgrounding

- ps -aMx will show priority – background is 4 or less

```
anthony     1547  ??  0.0 S   4T   0:00.01  0:00.02 <process name>
            1547      0.0 S   4T   0:00.00  0:00.00
            1547      0.0 S   4T   0:00.00  0:00.00
```

- spindump – look for throttle_lowpri_io frame

```
Process:          accountsd [242]
Importance:       Adaptive, Background Priority
```

- taskpolicy

```
$ taskpolicy –b <your command>
```

# Debugging Backgrounding

- ps -aMx will show priority – background is 4 or less

```
anthony     1547  ??  0.0 S   4T   0:00.01  0:00.02 <process name>
            1547      0.0 S   4T   0:00.00  0:00.00
            1547      0.0 S   4T   0:00.00  0:00.00
```

- spindump – look for throttle_lowpri_io frame

```
Process:         accountsd [242]
Importance:      Adaptive, Background Priority
```

- taskpolicy

```
$ taskpolicy -b <your command>
```

- fs_usage

```
13:02:43.124405 PgIn[AT3] D=0x022696e8 B=0x20000 /dev/disk1 0.000532 W
mds_stores.90196
```

# Simulating Constrained Systems

- Use boot-args to limit amount of available ram

```
sudo nvram boot-args="maxmem=2048"
```

- Use an external thunderbolt drives to simulate drive speeds
- Use Instruments preferences to limit number of CPUs

# More Information

**Paul Danbold**
Core OS Evangelist
danbold@apple.com

**Dave Delong**
Developer Tools Evangelist
delong@apple.com

**Apple Developer Forums**
http://devforums.apple.com

# Related Sessions

| | | |
|---|---|---|
| Maximizing Battery Life on OS X | Mission<br>Tuesday 11:30AM | 🟩 |
| Efficient Design with XPC | Russian Hill<br>Tuesday 2:00PM | 🟩 |
| Improving Power Efficiency with App Nap | Pacific Heights<br>Wednesday 10:15AM | 🟦 |
| Optimizing Drawing and Scrolling on OS X | Marina<br>Wednesday 3:15PM | 🟦 |
| Energy Best Practices | Marina<br>Thursday 10:15AM | 🟩 |
| Fixing Memory Issues | Nob Hill<br>Thursday 2:00PM | 🟧 |

# Labs

| | | |
|---|---|---|
| Power and Performance for OS X Apps | Core OS Lab A<br>Wednesday 9:00AM | 🟩 |
| Web Content Optimization Lab | Media Lab A<br>Wednesday 10:15AM | 🟪 |
| Cocoa and Foundation Lab | Frameworks Lab A<br>Wednesday 11:30AM | 🟦 |
| Instruments and Performance Lab | Tools Lab B<br>Thursday 3:15PM | 🟧 |
| Power and Performance for OS X Apps | Tools Lab A<br>Thursday 4:30 PM | 🟧 |

# Summary

- Regularly profile and optimize
- Measure both your app’s performance and resource efficiency
- Remember that your users may have very different systems
- Ensure your app is a good citizen

WWDC2013