Sep 23, 2025
Kernel Recipes’25, Paris

Overcoming Observer Effects in Memory Management with DAMON

SeongJae Park (SJ) <sj@kernel.org> <sjpark@meta.com>

https://damonitor.github.io

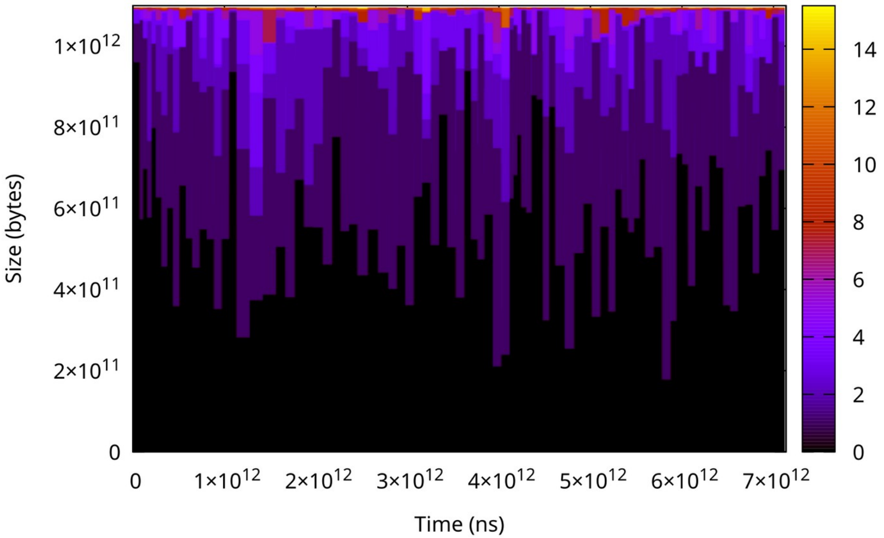
QR code generated from https://qr.io

Table of Contents

A Story:
Once Upon a Time,
In a Cloud Provider
Far, Far Away

An AWSome Team Started an Adventure for a Memory Auto-Scaling Service

- Motivation: Users ain't always need every byte of given VM's memory
- Idea: Dynamically adjust VM memory size to fit to only the *real* memory requirement
- Provider profit: Higher physical resource efficiency, accurate bill calculation
- User profit: Less cost, no performance degradation



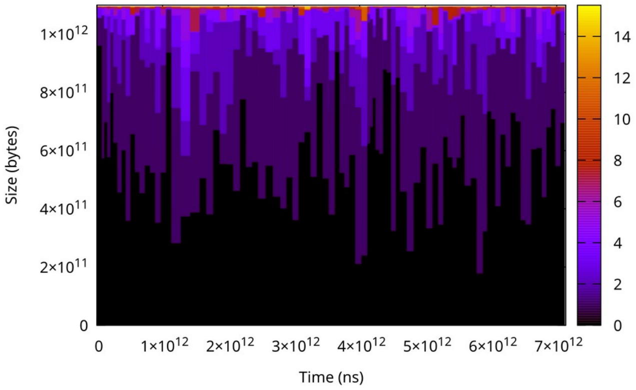
An AWSome Team Started an Adventure for a Memory Auto-Scaling Service

- Motivation: Users ain’t always need every byte of given VM’s memory
- Idea: Dynamically adjust VM memory size to fit to only the *real* memory requirement
- Provider profit: Higher physical resource efficiency, accurate bill calculation
- User profit: Less cost, no performance degradation

```
User                                  Service Provider

workload, min/max memory  ------->  | Fixed total ram, |
workload output, bill     <-------  | Flexible # VMs   |
```

An AWSome Team Started an Adventure for a Memory Auto-Scaling Service

- Motivation: Users ain’t always need every byte of given VM’s memory
- Idea: Dynamically adjust VM memory size to fit to only the *real* memory requirement
- Provider profit: Higher physical resource efficiency, accurate bill calculation
- User profit: Less cost, no performance degradation




```
User                                     Service Provider

workload, min/max memory  ------->   | Fixed total ram, |
workload output, bill     <-------   | Flexible # VMs   |
```

The Quest: Knowing *Real* Memory Requirement

- Allocated memory != Real (or, critical) memory requirements
- Major challenge: Overhead and Accuracy
- No good solution was available back then (<Linux 5.15 era)

```
$ # damo monitor --report_type holistic $(pidof $MY_WORKLOAD)
[...]
# Memory Footprints Distribution
percentile                0                25                50                75               100
       wss       13.539 MiB        13.754 MiB        15.293 MiB        16.605 MiB        16.605 MiB
       rss      105.102 MiB       105.102 MiB       105.102 MiB       105.102 MiB       105.102 MiB
       vsz      108.277 MiB       108.277 MiB       108.277 MiB       108.277 MiB       108.277 MiB
  sys_used      943.090 MiB       943.090 MiB       943.090 MiB       943.090 MiB       943.090 MiB
[...]
```

User Meets Kernel

- A kernel programmer in Dresden was looking for users of their new kernel feature
- The feature was advertised as what the service team was looking for
- They eventually met and co-developed the service with the kernel feature

for such memory management. It is designed with some key mechanism (refer to Design for the detail) that make it

- accurate (the monitoring output is useful enough for DRAM level memory management; It might not be appropriate for CPU Cache levels, though),
- light-weight (the monitoring overhead is low enough to be applied online), and
- scalable (the upper-bound of the overhead is in constant range regardless of the size of target workloads).

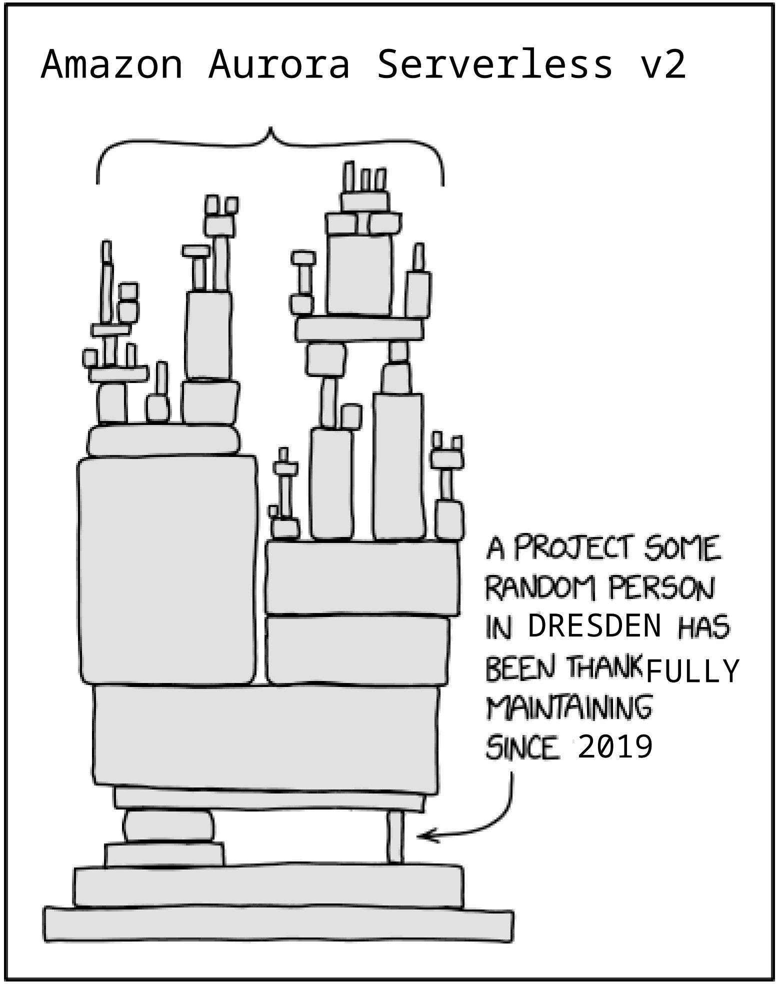
<Image captured from the feature’s project website>

They Lived Happily Ever After (So Far)

- The service has successfully launched:
  Amazon Aurora Serverless v2

- The subsystem has merged into Linux 5.15: DAMON

- DAMON continues its evolution for more users



Retrieved and modified from https://xkcd.com/2347/

Observer Effects in Memory Management

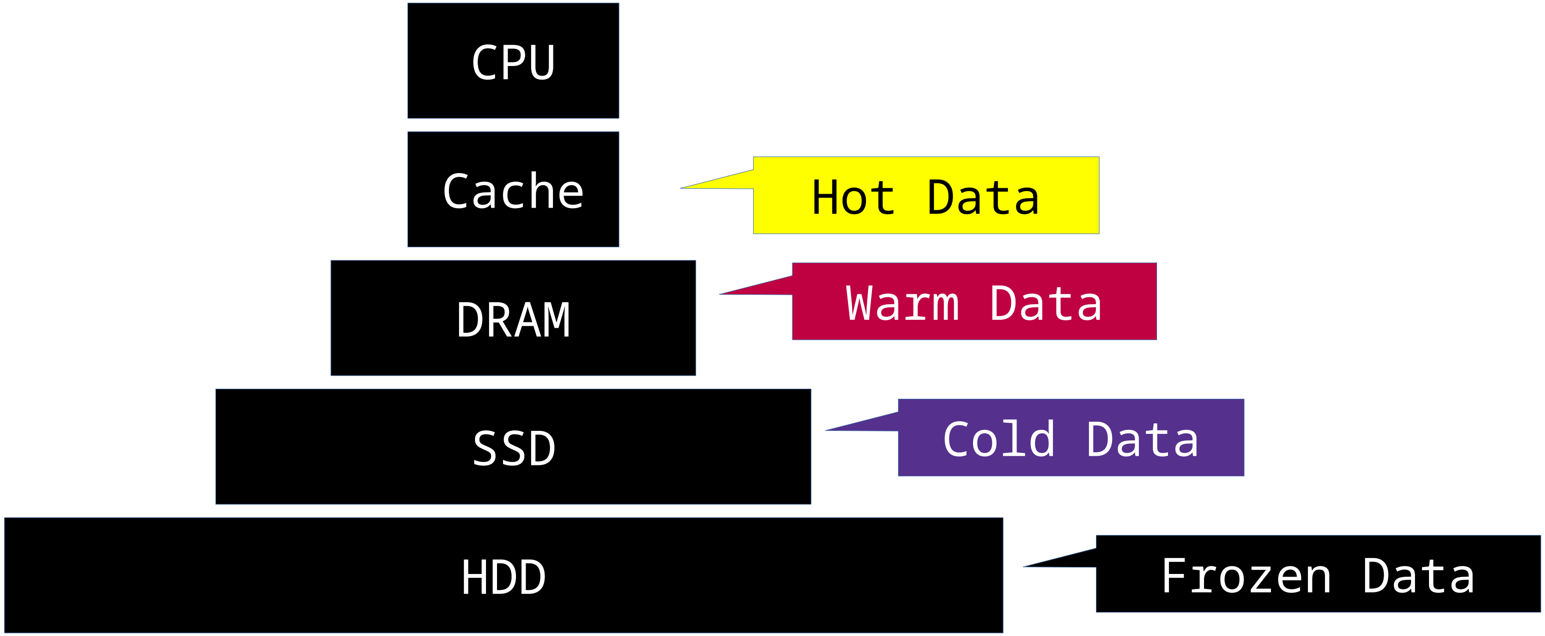
Memory: What It Is, and Why Limited?

- Goal of Computers: processing data
- Memory: Medium for storing/loading data
- Consistent Trend: Exploding size of data
- Turing Machine Idea: Infinite memory
- Limitations of Physics ($E = mc^2$; m: mass of electrons on modern computers)
  - Speed of processor > Speed of memory
  - Physical memory cost $\propto$ Access speed and capacity of memory

“Everyone Has a Plan Until They Get Punched in the Mouth”,
Mike Tyson

H/W Solution for Memory Limitation: Hierarchical Memory System

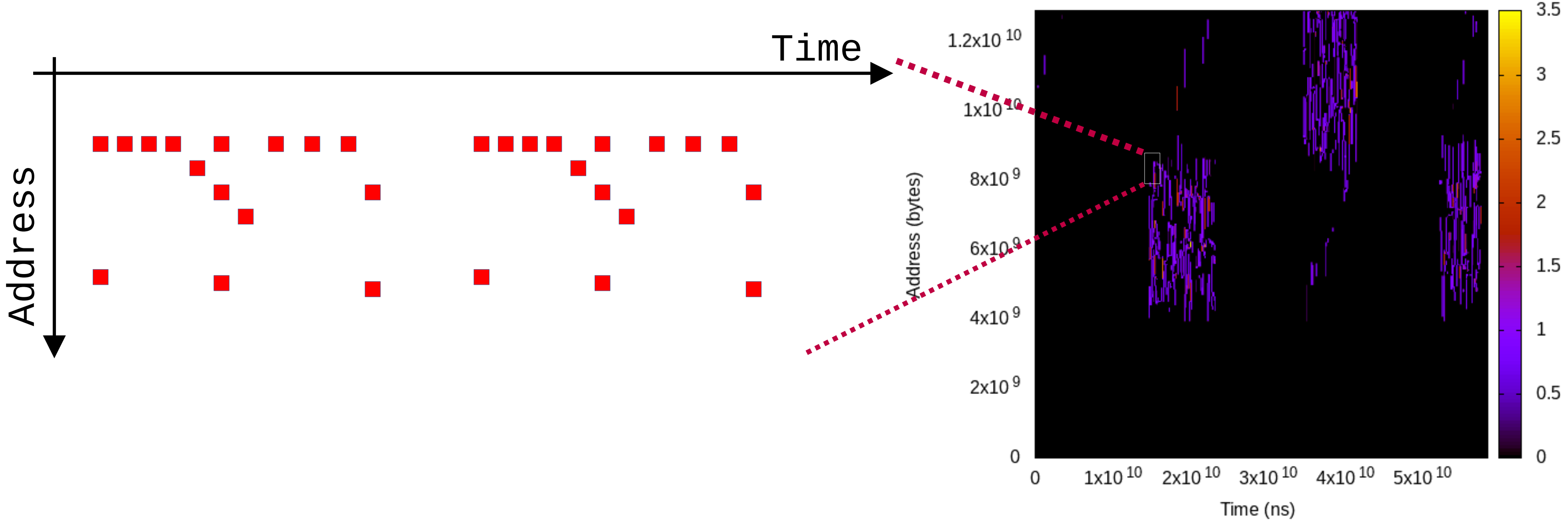
- Hierarchical memory: construct memory with different cost/performance devices
  - Fastest and smallest device on uppermost layer (nearest to the processor)
  - Put more frequently accessed (hot) data on upper layer
  - H/W cannot deal with all complicated access-aware management scenarios, though



S/W for Optimized Hierarchical Memory Management

- Goal: Keep hottest memory in uppermost layer of hierarchical memory
- OS Kernel (S/W) can optimize access-aware data placement for complicated cases
  - Evict cold data to lower layer (a.k.a reclaim, tiered-memory demotion)
  - Migrate hot data to upper layer (a.k.a NUMA balancing, tiered-memory promotion)
  - Sounds simple. Now, how it knows the access temperature (pattern)?

Data Accesses: Microscope Events on Space/Time of Memory



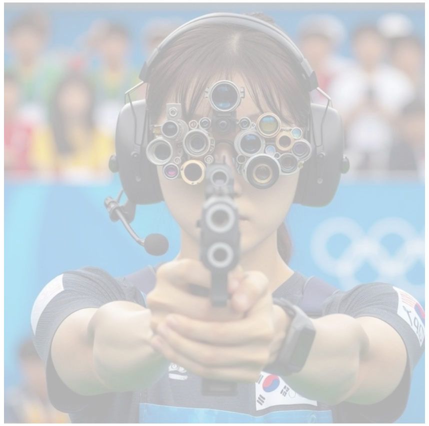
Observer Effects in Data Access Monitoring

- Ideal: Precise (every bit), Complete (every moment), Light (prod online)
- Plan: Record every access whenever it is made
- Bad reality: Observer effect is inevitable
  - Recording itself require memory access
  - Add monitoring-purpose memory access for each memory access
- Good reality: You Ain’t Gonna Need It
  - Even quantum physics PhD can live without knowing where each electron is at the moment
  - For memory management, a high level view can be enough

“Everyone Has a Plan Until They Get Punched in the Mouth”, Mike Tyson

Access Monitoring Approaches for Linux Memory Management

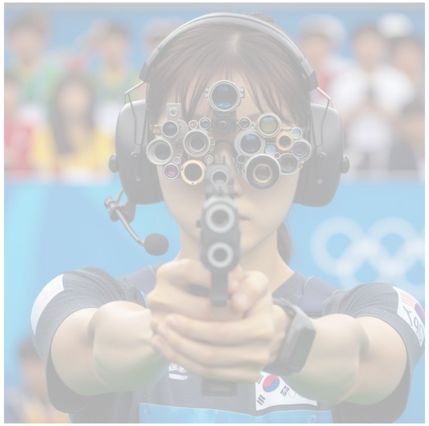
- Use non-ideal but practical mechanisms of two categories
- Developed for individual management mechanism
  - E.g., Pseudo-LRU and artificial page faults for reclaimation and NUMA balancing
  - Obscure, heuristic-based, but time-tested
- Developed for observable and holistic memory managements: DAMON



Images retrieved from https://visla.kr/article/etc/119021/ and https://x.com/DeepinJapanPod/status/1819569233124376815

Access Monitoring Approaches for Linux Memory Management

- Use non-ideal but practical mechanisms of two categories
- Developed for individual management mechanism
  - E.g., Pseudo-LRU and artificial page faults for reclaimation and NUMA balancing
  - Obscure, heuristic-based, but time-tested
- Developed for observable and holistic memory managements: DAMON



Images retrieved from https://visla.kr/article/etc/119021/ and https://x.com/DeepinJapanPod/status/1819569233124376815

Access Monitoring Approaches for Linux Memory Management

- Use non-ideal but practical mechanisms of two categories
- Developed for individual management mechanism
  - E.g., Pseudo-LRU and artificial page faults for reclaimation and NUMA balancing
  - Obscure, heuristic-based, but time-tested
- Developed for observable and holistic memory managements: DAMON

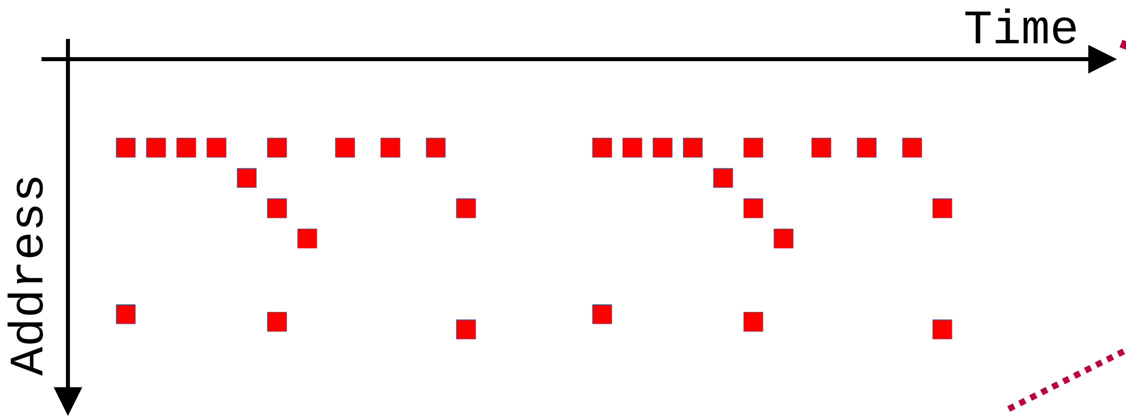
The topic of this talk

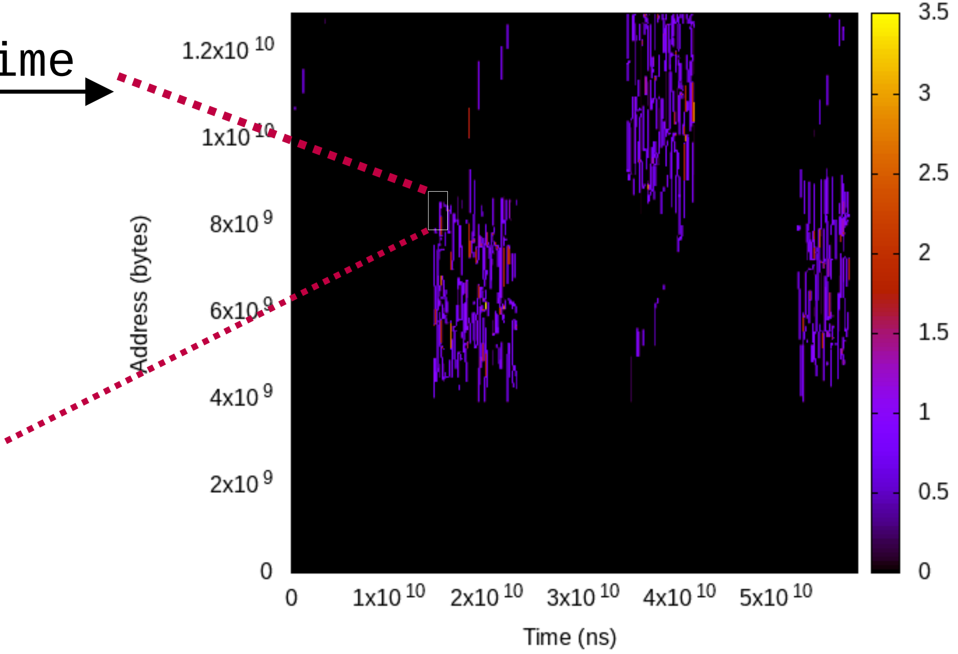
Images retrieved from https://visla.kr/article/etc/119021/ and https://x.com/DeepinJapanPod/status/1819569233124376815

How DAMON Handles The Observer Effects: 0. Goal and Challenges

DAMON Goal: Access Observability for Holistic Memory Management

- Draw address-time access map or reasonably equal form of information





DAMON Challenges: Overhead

- Time Overhead
  - For generating each snapshot of the map
  - O(memory size)
- Space Overhead
  - For saving the entire access events map
  - O(memory size * total monitoring time)
- Memory size and monitoring time can be arbitrarily huge

How DAMON Handles The Observer Effects: 1. Region-based Sampling

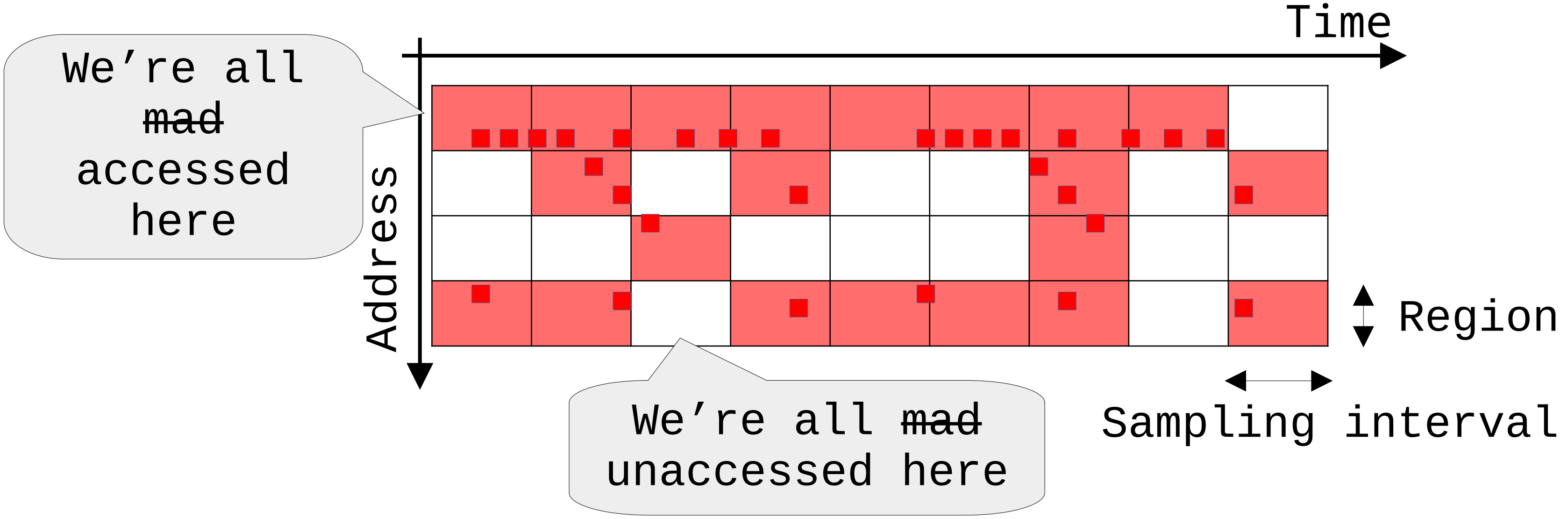
Region: Access Monitoring Unit for DAMON

- Defined as a *reasonably-atomic* unit of data access
  - A sub-area of the memory's space-time
  - A collection of adjacent elements that having similar access pattern
- By the definition, access check of one element per region is enough
- e.g., "This page is accessed within last 1 second; I checked a cacheline in it"

```
$ cat wonder_region_1
We're all ~~mad~~ [un]accessed here
```

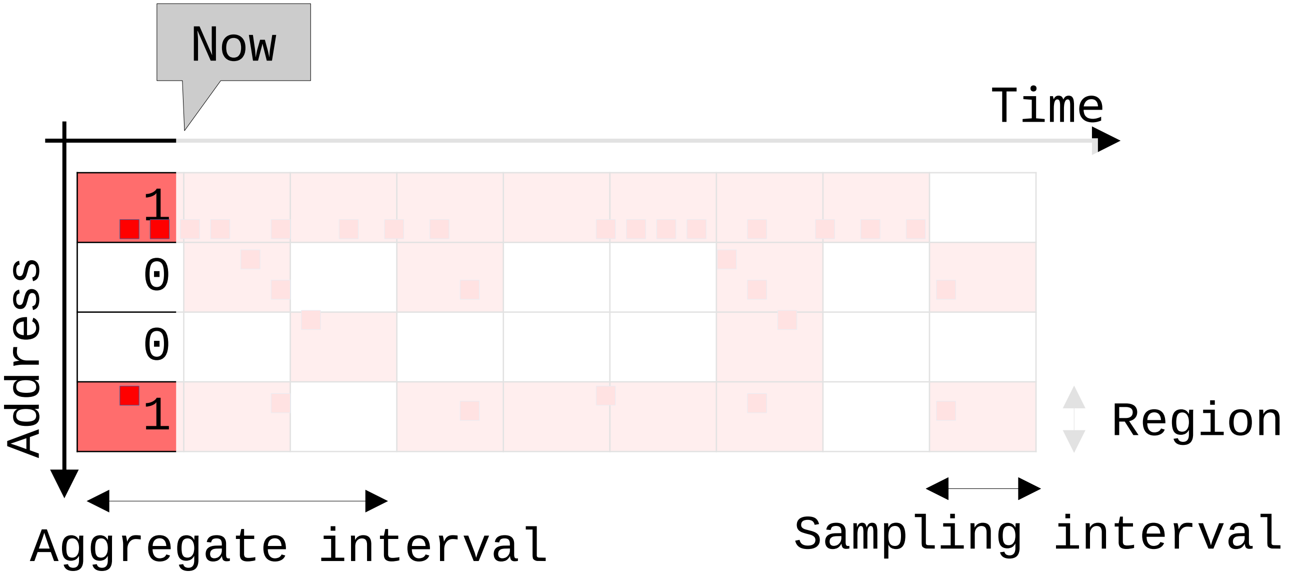
Fixed Space/Time Granularity, Boolean Access Frequency

- Time overhead: “`memory size /` ***`space granularity`***”
- Space overhead: “`time overhead * monitoring time /` ***`time granularity`***”
- Overhead is reducible and controllable
- Ruled by memory size and monitoring time, finding best granularity is challenging



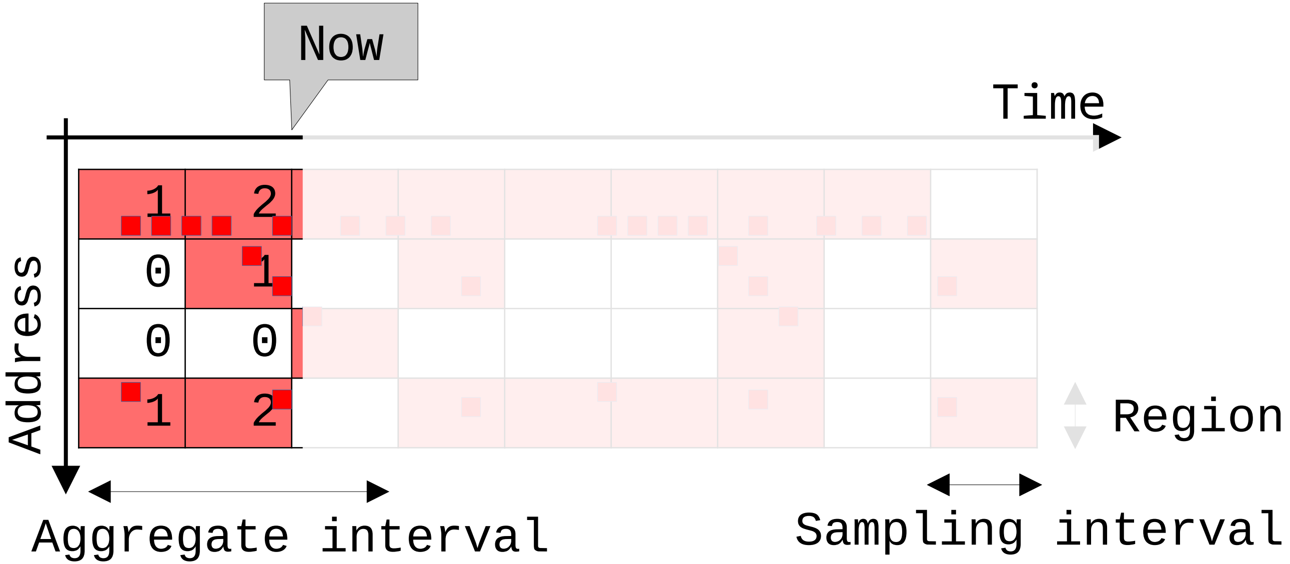
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate (sampled) access check results via per-region counter



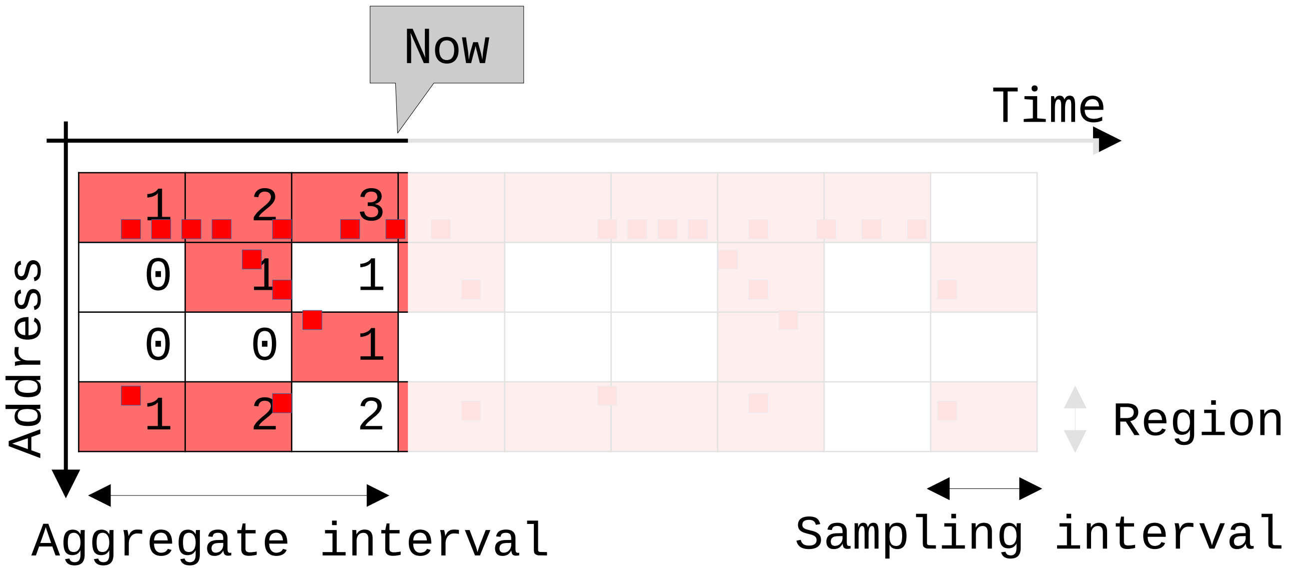
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate (sampled) access check results via per-region counter



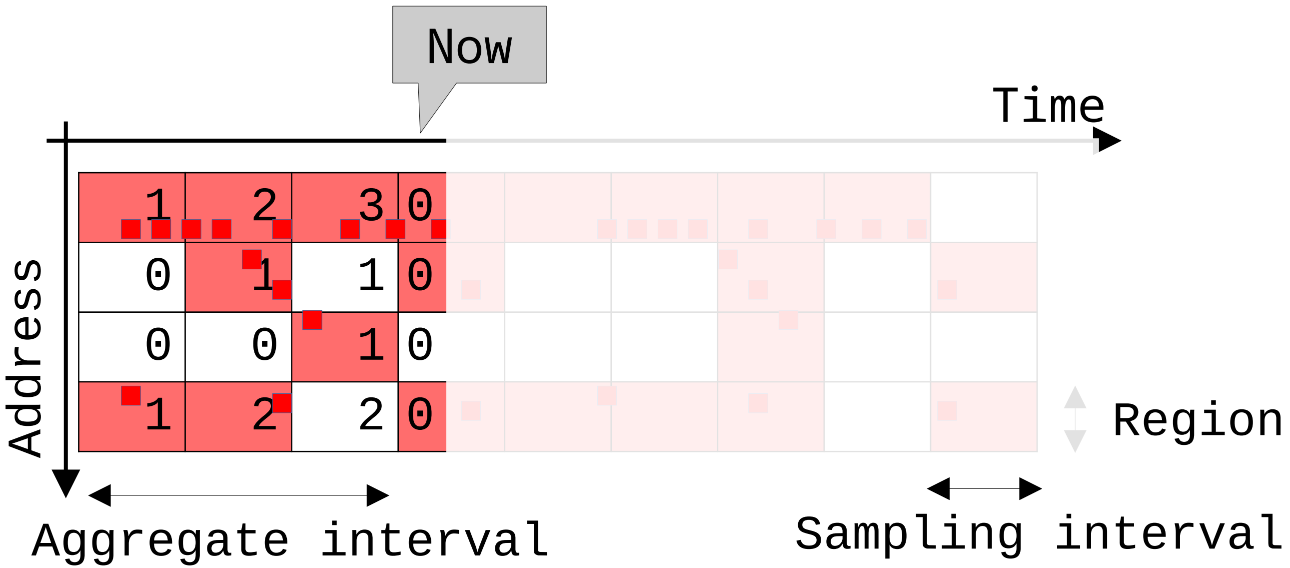
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate (sampled) access check results via per-region counter



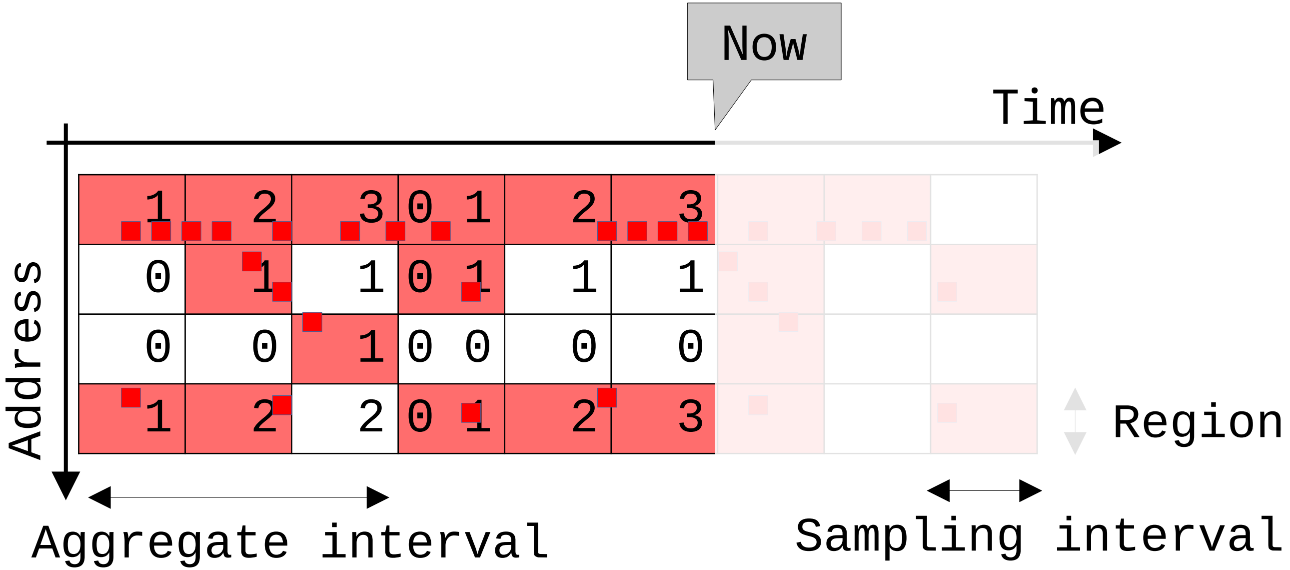
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate (sampled) access check results via per-region counter



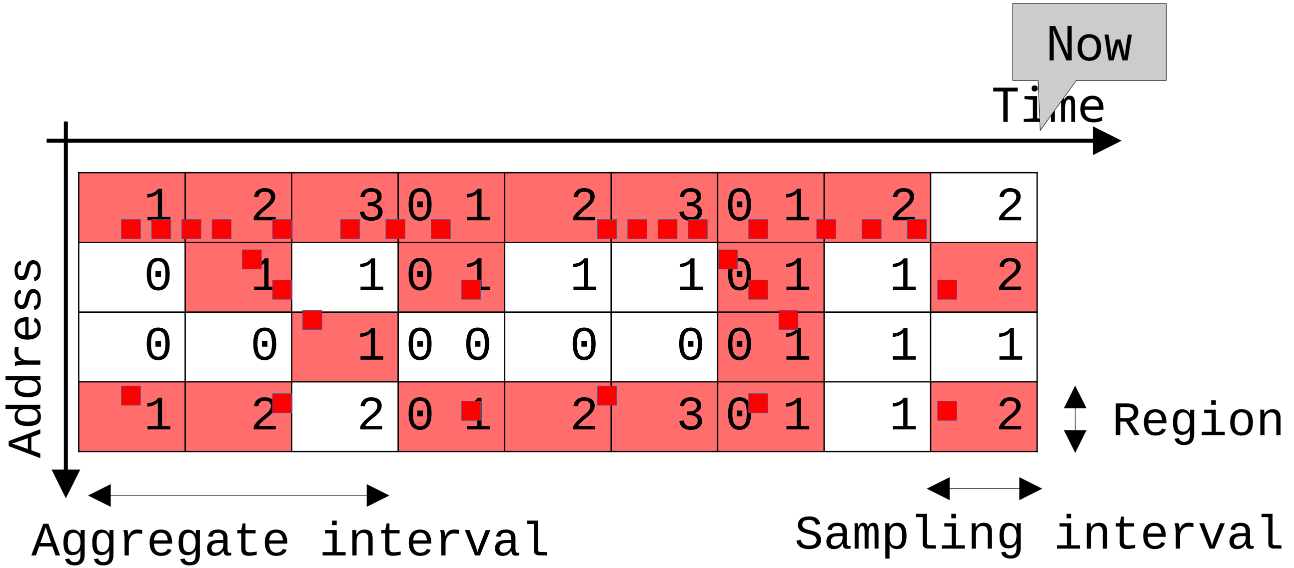
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate (sampled) access check results via per-region counter



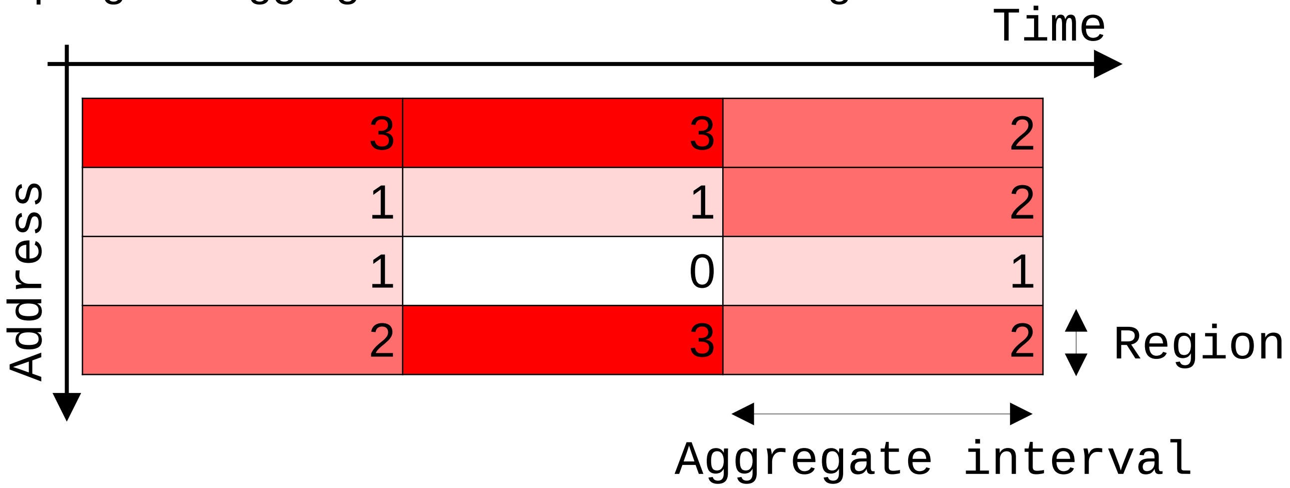
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate (sampled) access check results via per-region counter



Fixed Space/Time Granularity, <=N Access Frequency

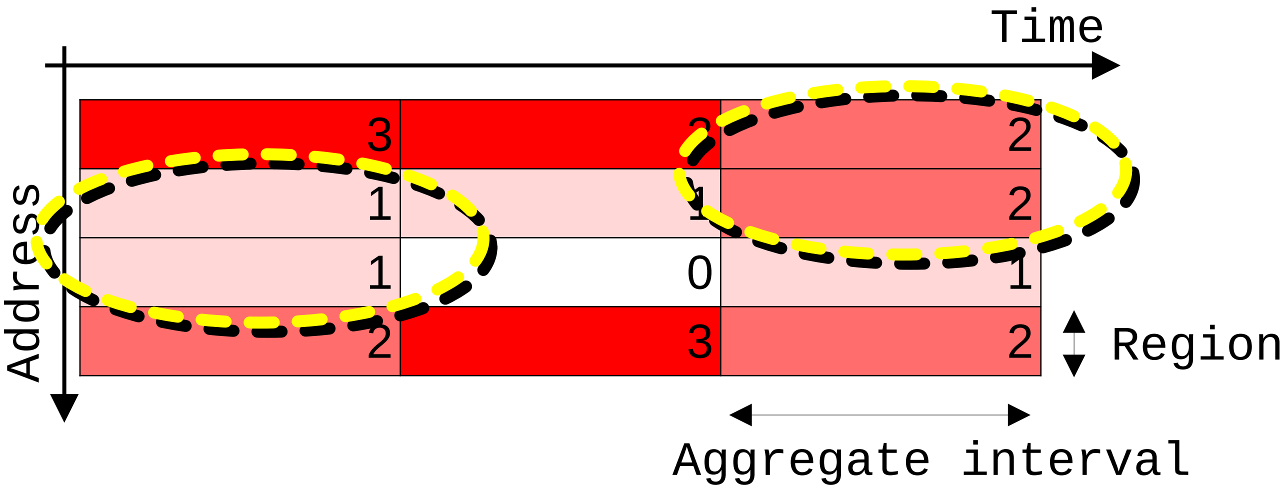
- Accumulate access checks via per-region counter
- Reduce space overhead to “1/***N***”
- Still, O(memory size * total monitoring time)
- Optimum sampling and aggregation intervals? Following slides will cover.



How DAMON Handles The Observer Effects: 2. Self-tuned Region Space

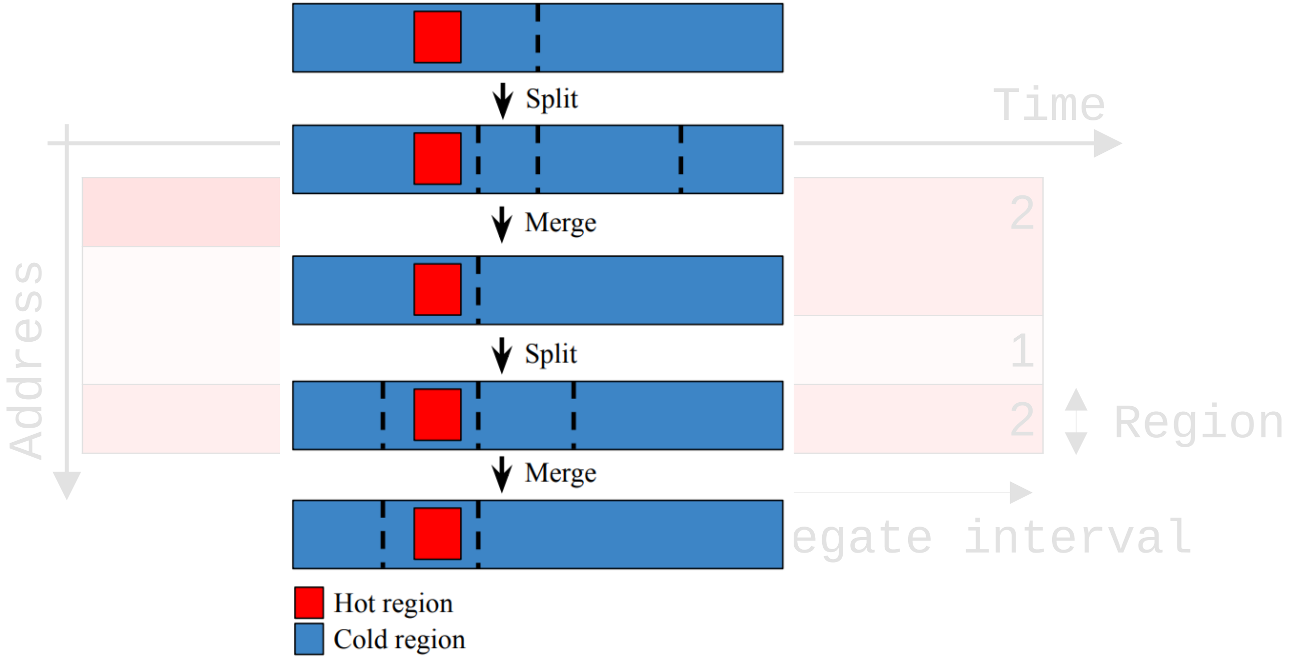
Problems of Fixed Space Granularity

- Adjacent regions of similar hotness are wastes
- Restrict fine-grained space monitoring



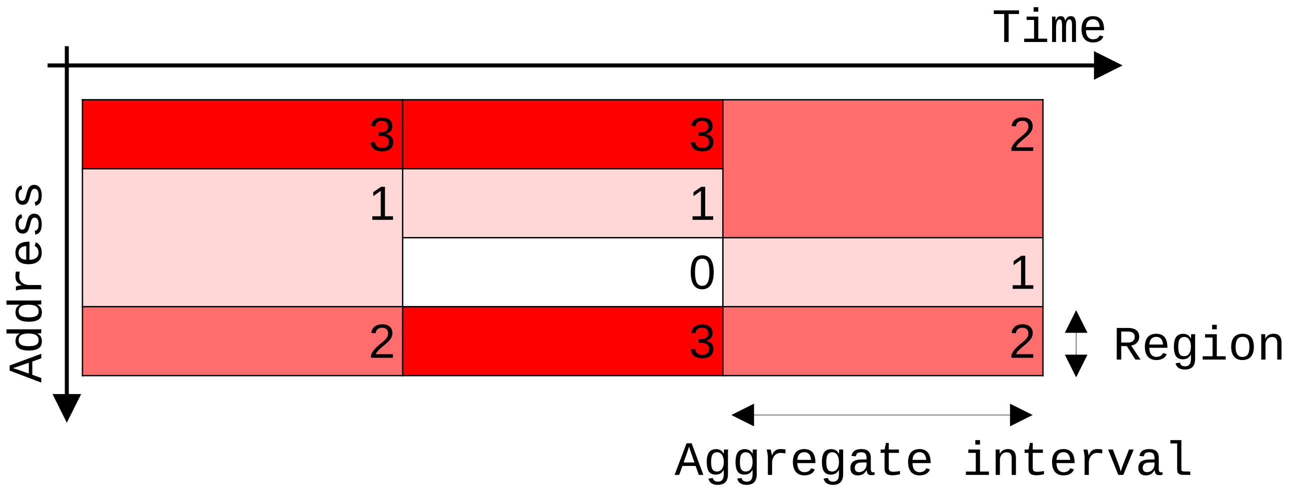
Auto-tuned Dynamic Space Granularity: Mechanisms (1/2)

- Repeat merging the wasteful regions and randomly splitting regions
  - The number of region == number of different access patterns
- Let user set min/max number of total regions (10 and 1000 are defaults and recommended)



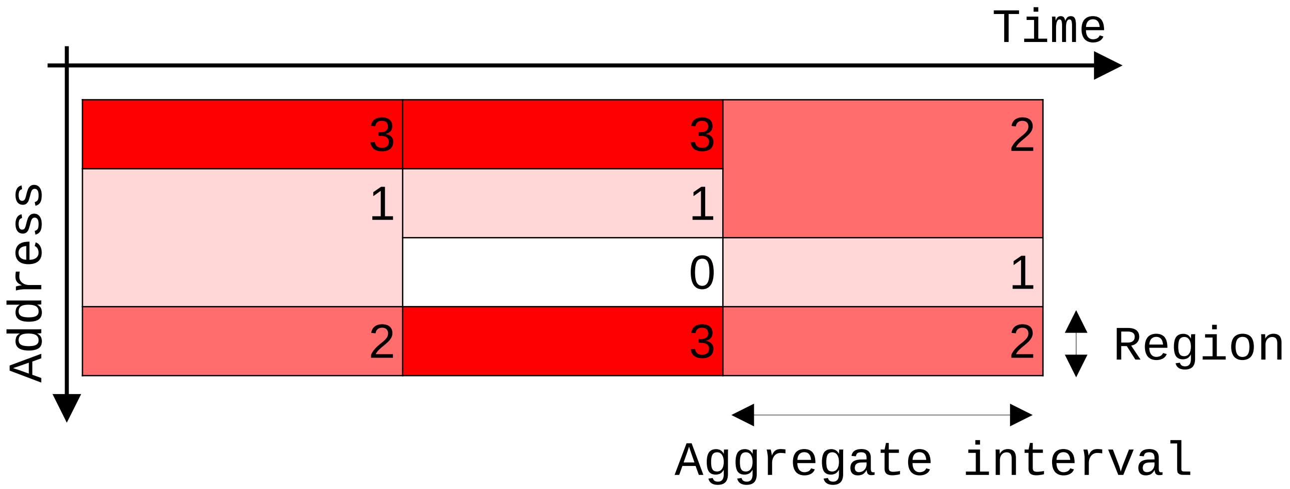
Auto-tuned Dynamic Space Granularity: Mechanisms (2/2)

- Repeat merging the wasteful regions and randomly splitting regions
  - The number of region == number of different access patterns
- Let user set min/max number of total regions (10 and 1000 are defaults and recommended)



Auto-tuned Dynamic Space Granularity: Overhead/Accuracy

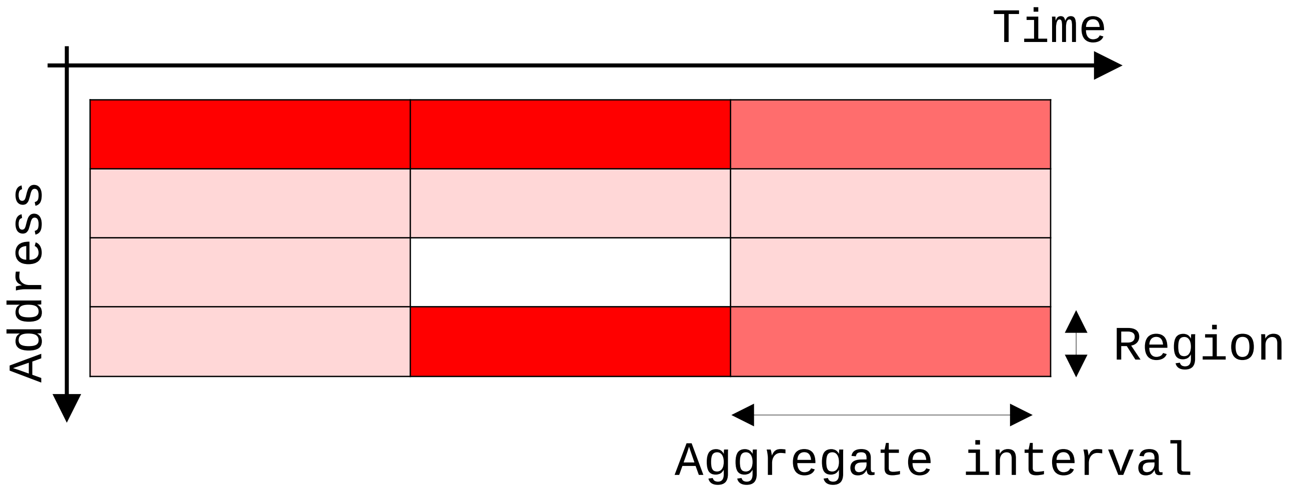
- Time overhead: min(different access patterns, max number of regions)
  - No more ruled by memory size, fully controlled and auto-tuned
- Accuracy: best-effort high
  - Auto-tuned dynamic granularity can find accesses to small memory area



How DAMON Handles The Observer Effects: 3. Self-tuned Region Time

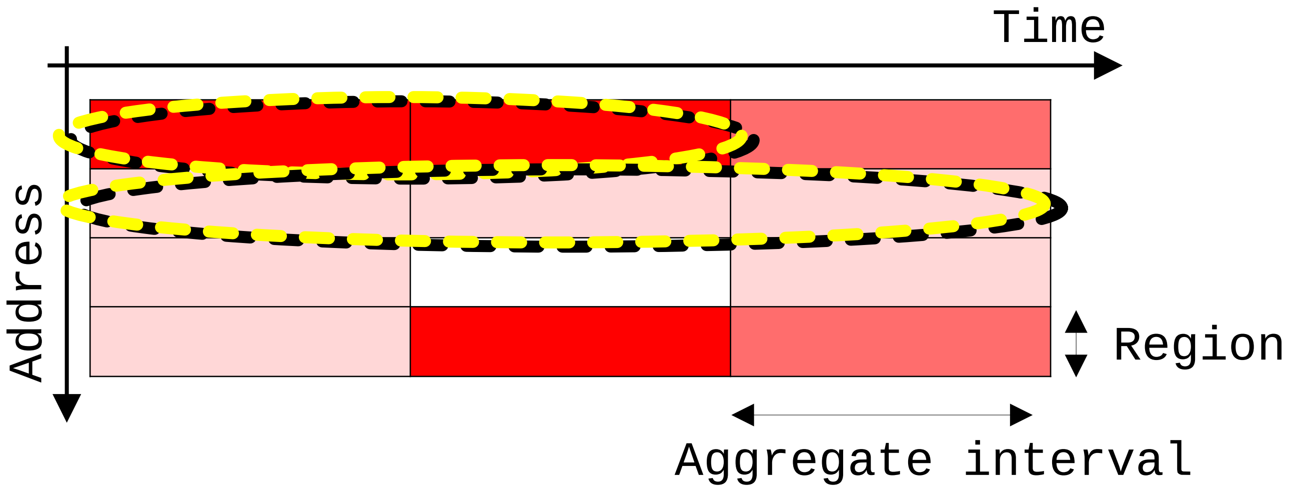
Problems of Fixed Time Granularity Regions (1/2)

- The definition of regions is not only about space, but also about time



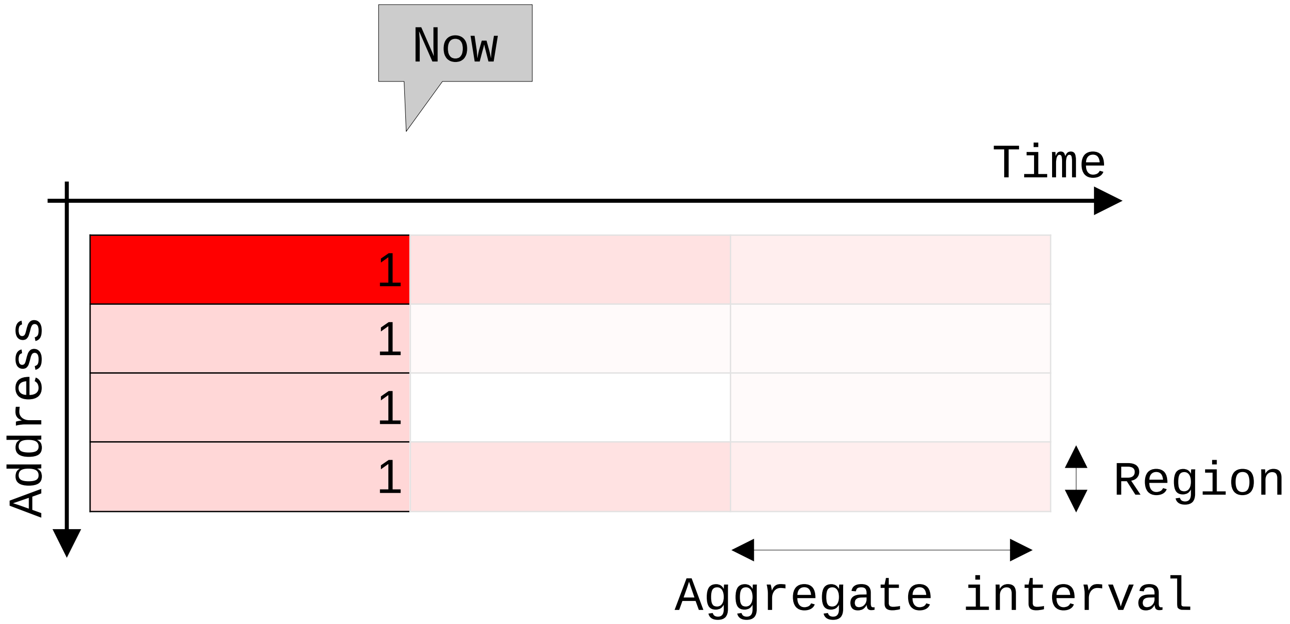
Inefficiency of Fixed Time Granularity Regions (2/2)

- The definition of regions is not only about space, but also about time
- Multiple time-adjacent regions of similar hotness: only waste



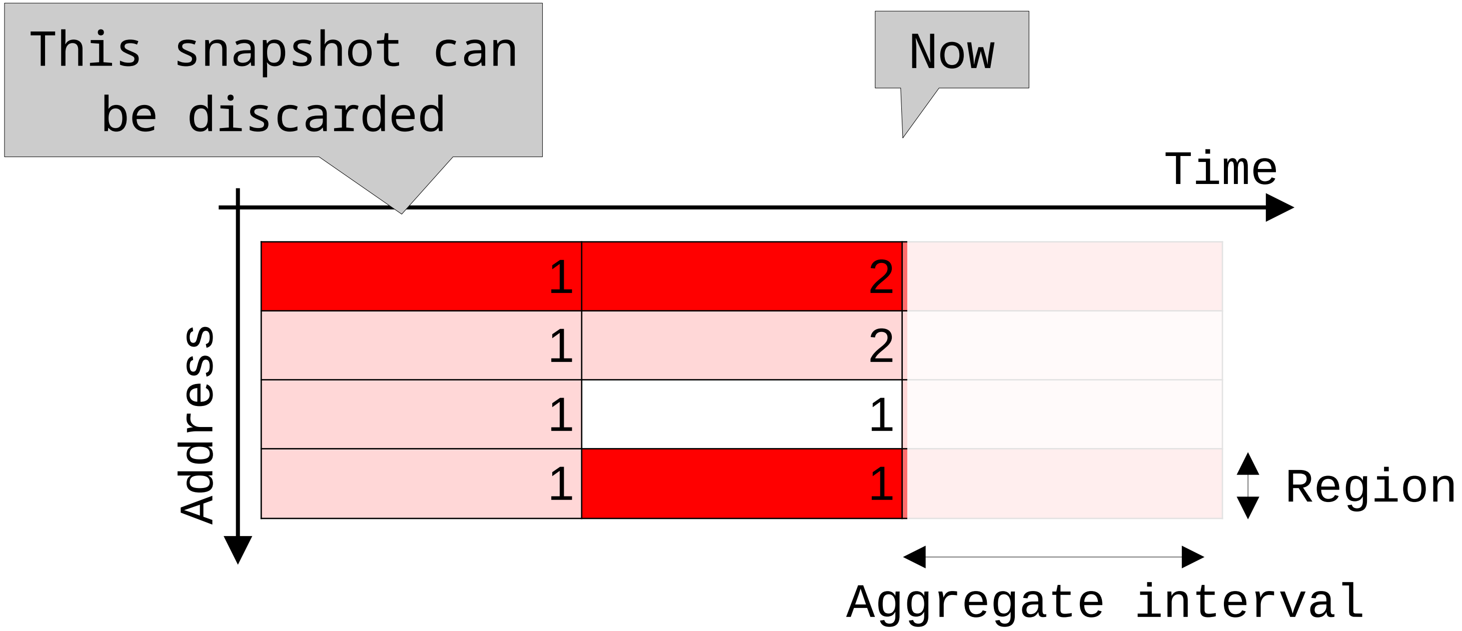
Dynamic Time Granularity (1/3)

- Count how long the hotness has kept
- Snapshot contains history of useful length



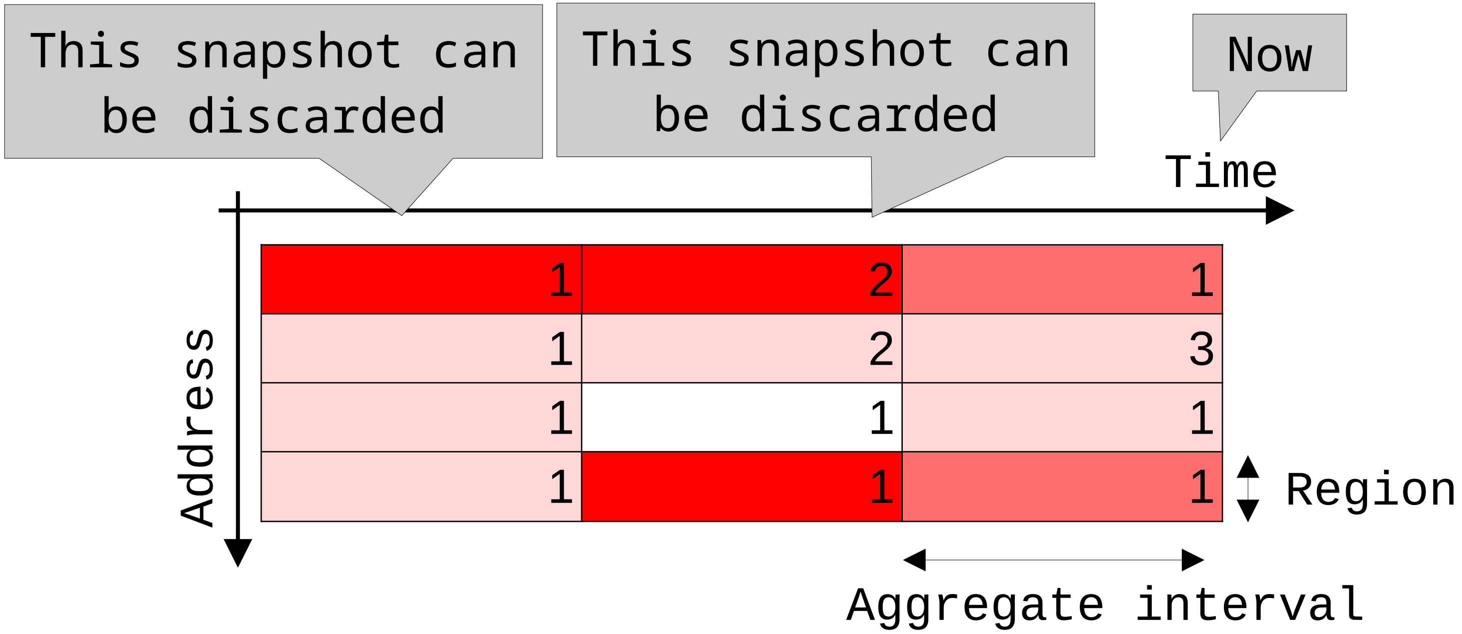
Dynamic Time Granularity (2/3)

- Count how long the hotness has kept
- Snapshot contains history of useful length



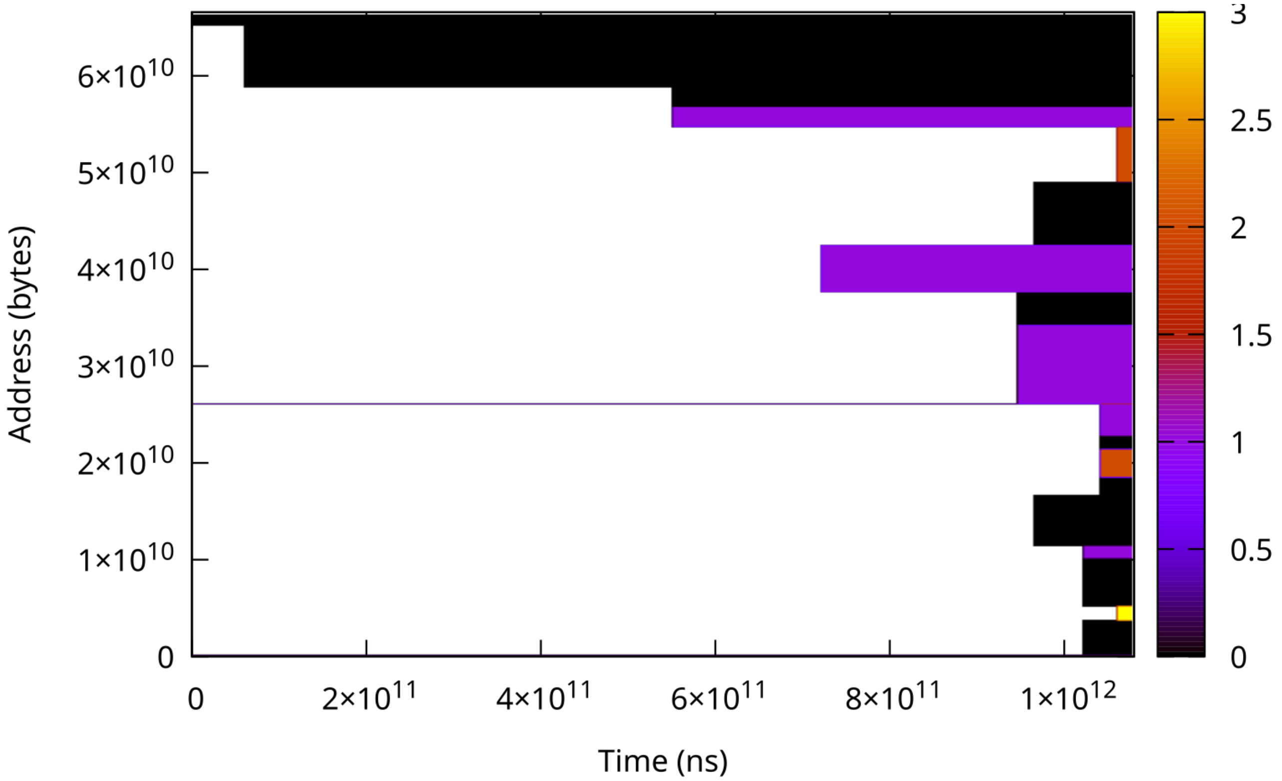
Dynamic Time Granularity (3/3)

- Count how long the hotness has kept
- Snapshot contains history of useful length



Snapshot: The Output of DAMON

- O(max_nr_regions) time/space overhead
- Both time/space overheads are not ruled by memory size/monitoring time



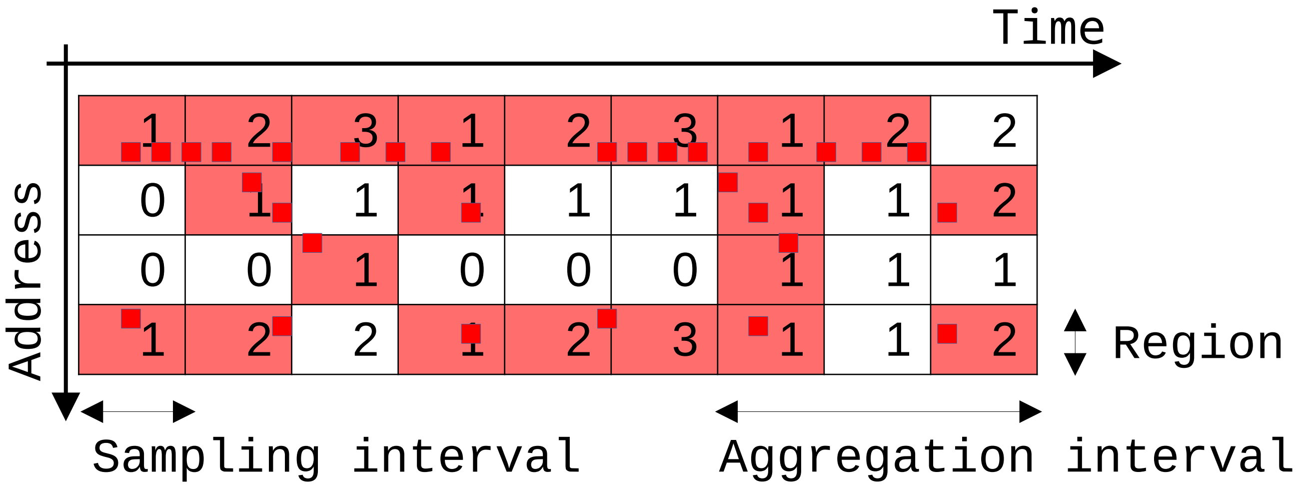
How DAMON Handles The Observer Effects: 4. Monitoring Intervals Auto-tuning

FAQ: DAMON output looks only cold, only hot, or just random

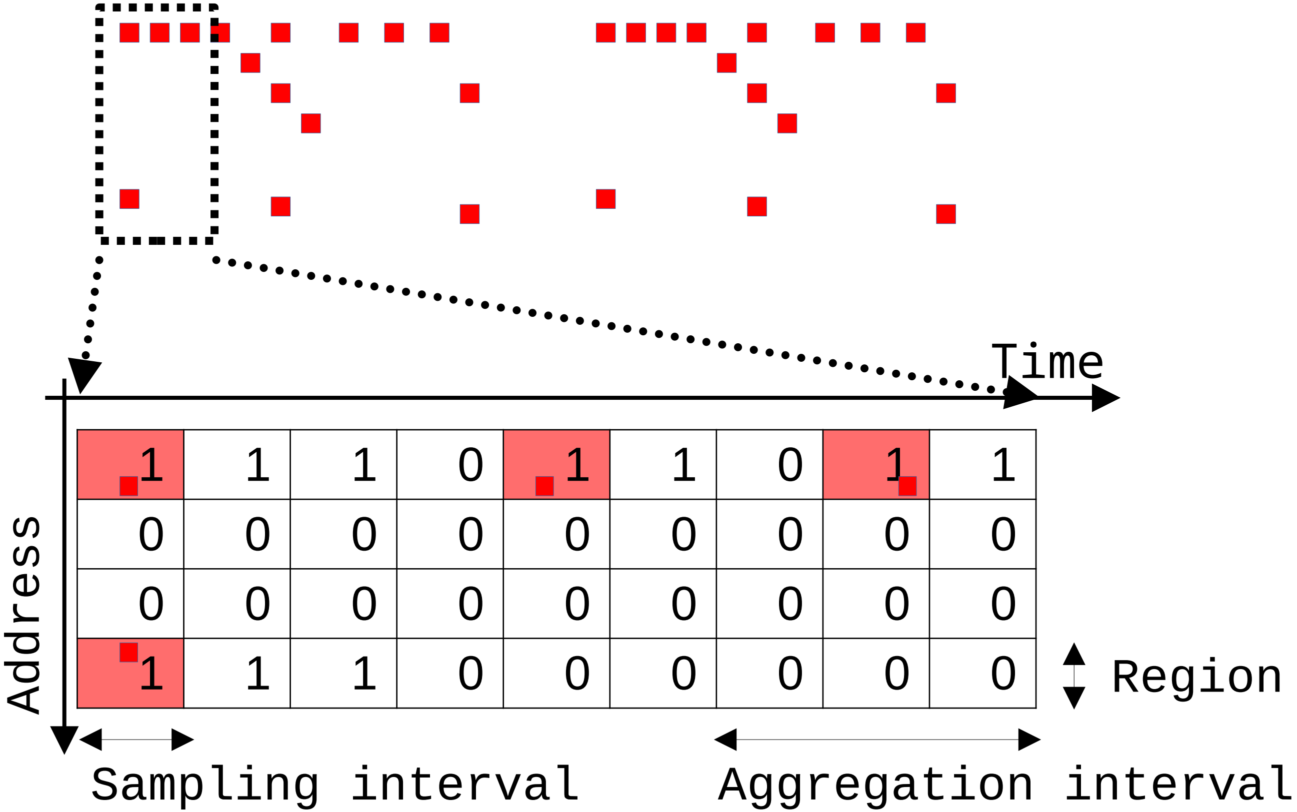
- Frequent Answer: Have you tuned the monitoring intervals?
  - IOW, the default intervals (5ms sampling, 100ms aggregation) are not really suggested ones

If Intervals Are Appropriate: Meaningful Hot/Cold Regions

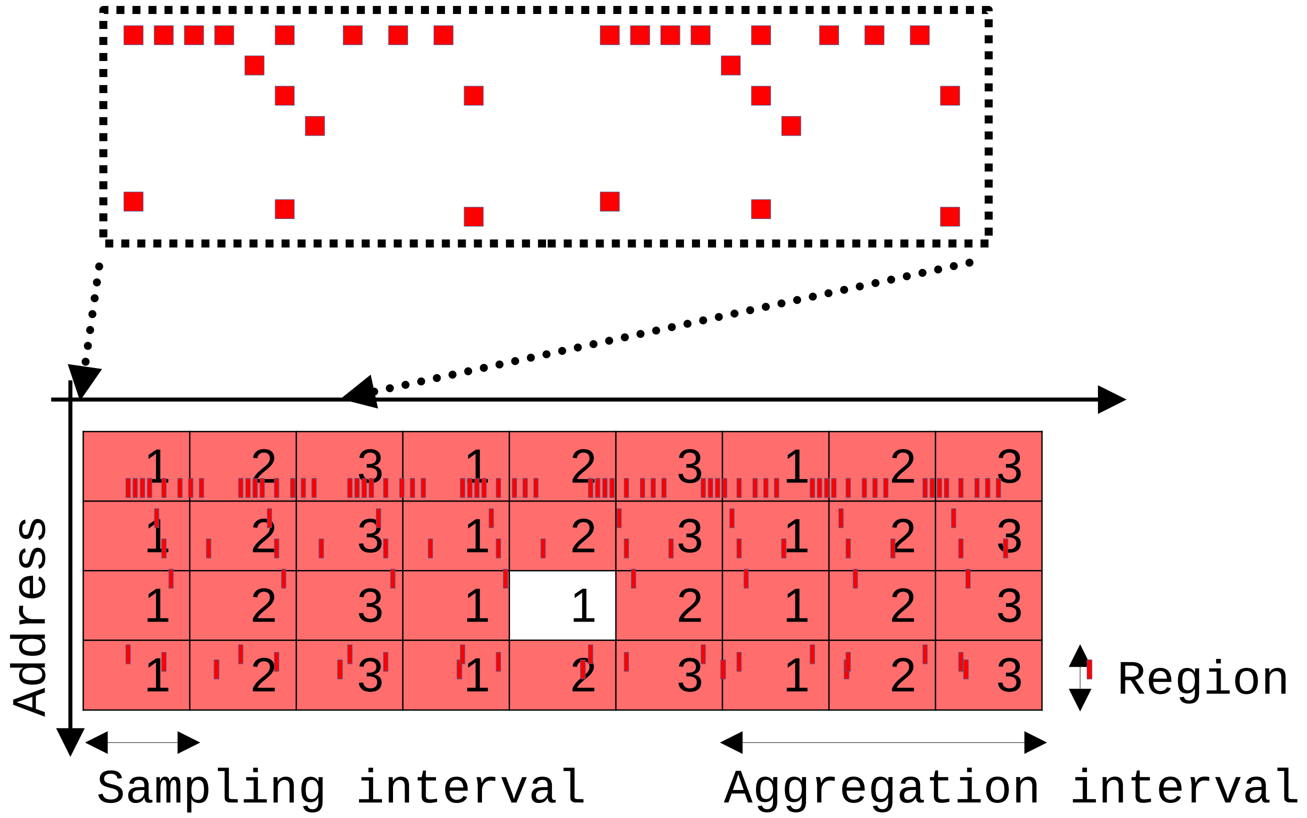
- Meaningful enough to make some memory management decisions



If Intervals Are Too Short: Everything Looks Cold

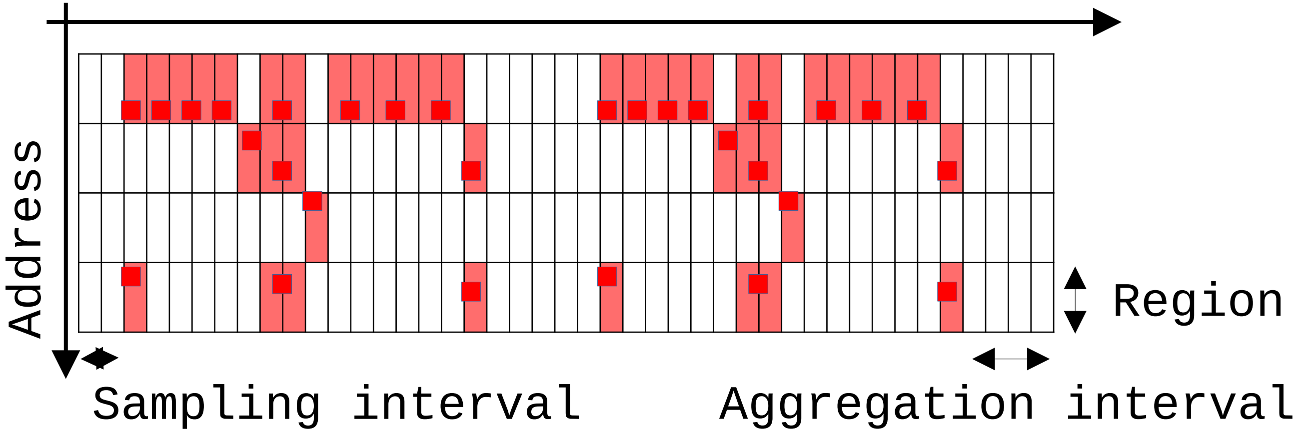


If Intervals Are Too Long: Everything Looks Hot



If Sampling:Aggregation Interval ratio is Too Low: Meaningless Samples

- Most sampling returns “negative”: unnecessary CPU cycle waste
- On large systems, sampling quality can also degrade
  - Not enough time for workloads to leave footprints



Aimed Monitoring Output-oriented Intervals Auto-tuning

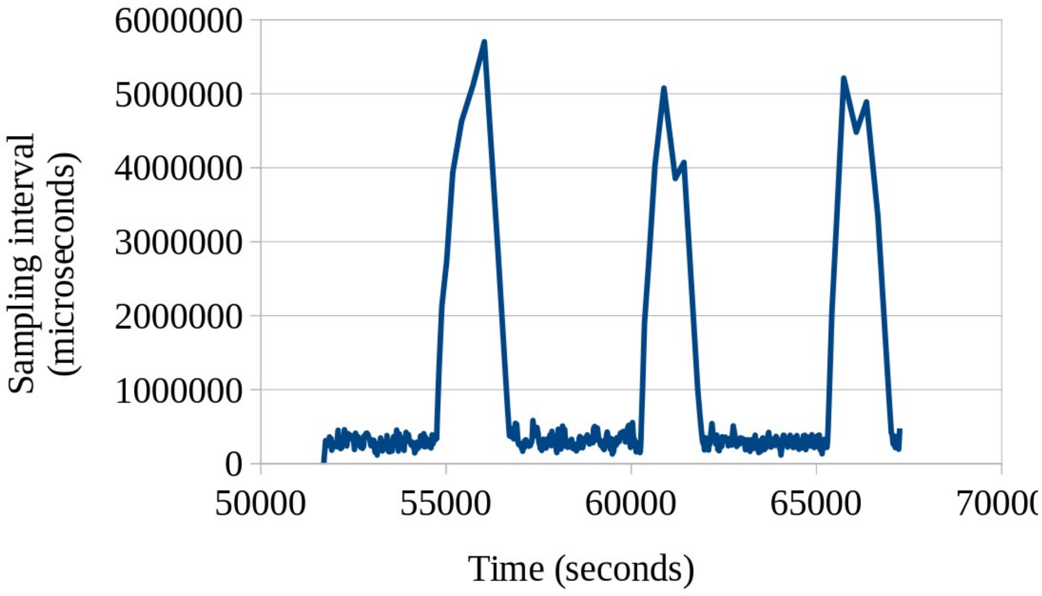
- Change Question: How to do? (mechanism) → What to achieve? (final goal, policy)
- Let users specify
  - Desired amount of access events to capture in each snapshot
  - Minimum and maximum sampling intervals
- Find sampling/aggregation intervals for the desire using a feedback loop
  - If less than desired events are captured in current snapshot, increase intervals
  - If more than desired events are captured in current snapshot, decrease intervals
  - Min/max sampling intervals ensure auto-tune goes no too long

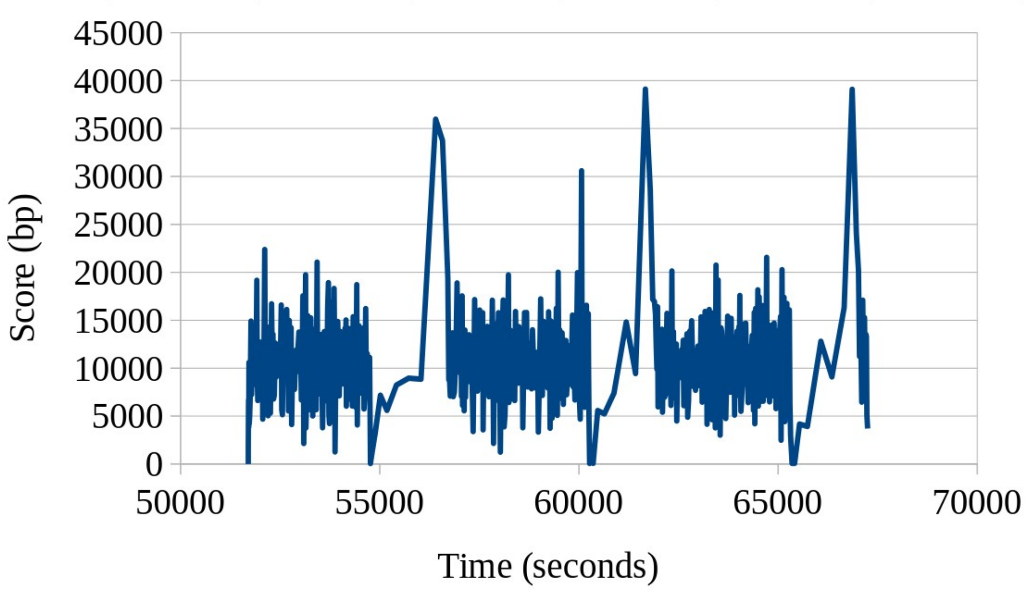
Monitoring Intervals Auto-tuning Parameters

- Parameters for parameters auto-tuning, but easy to set
- Suggestion
  - Desired access events per snapshot: 4% of maximum events that can be captured in snapshot
    - Applies Pareto principle (80:20 rule) twice, assume to capture 80% * 80% = 64% real access
  - Min/max sampling intervals: 5ms and 10s
  - Sampling:aggregation intervals ratio: 1:20
    - Only a few different actions are required, 20 is high enough

Intervals Auto-tuning on a Real-world Server Workload

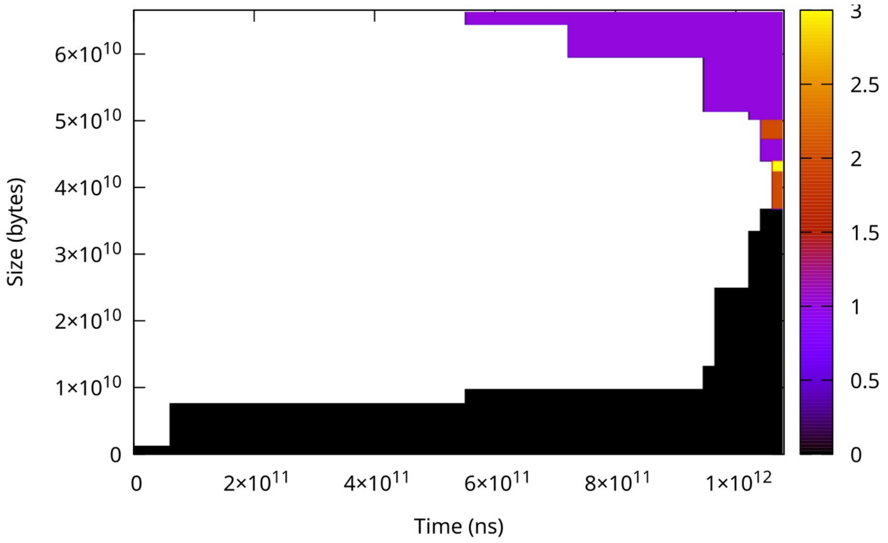
- Sampling interval and tuning score continuously change, and converge for given situation
  - Sampling interval converges to 370ms under usual load, ~4-5 seconds under light load
  - Tuning score converges to the goal (10,000 bp)

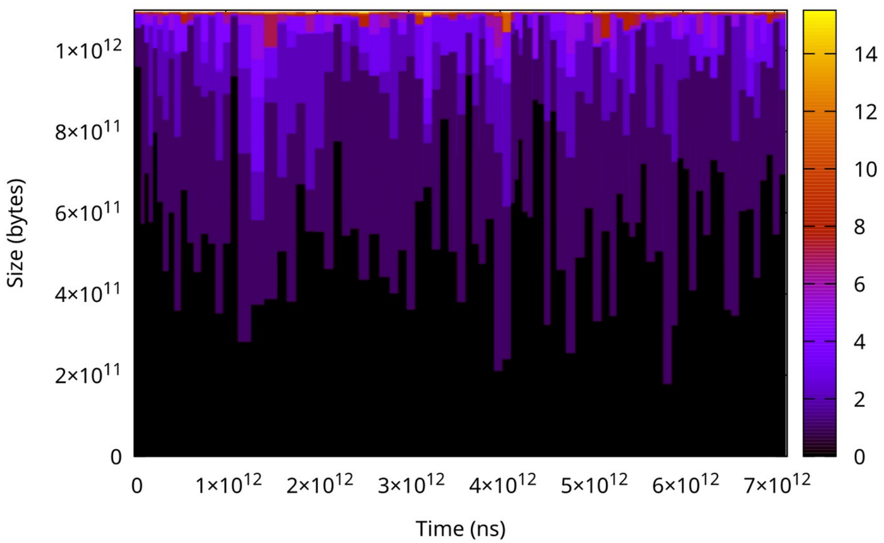




Intervals Auto-Tuning on Real World Server Workloads

- Meaningful access patterns found on three different workloads including 1 TiB memory size workload
- 0.0% CPU time consumed for the monitoring

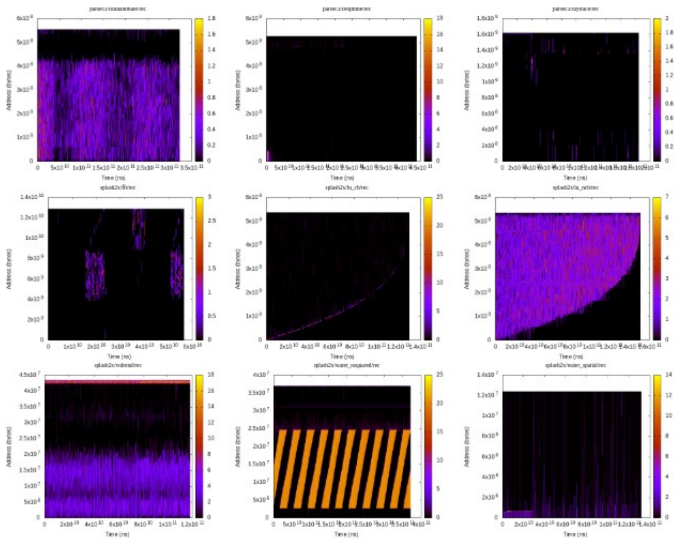




How DAMON Handles The Observer Effects: 5. Access-aware Memory Management

Utilizing DAMON for Access-aware Memory Management

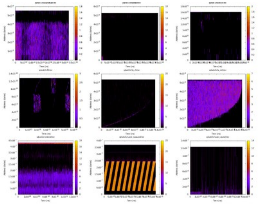
- Profiling (e.g., GIF demo link)
  - For finding rooms to improve, e.g., capacity planning
- Profiling-guided Optimizations
  - Could be done on both offline and online
- Why not let kernel just (transparently) works?



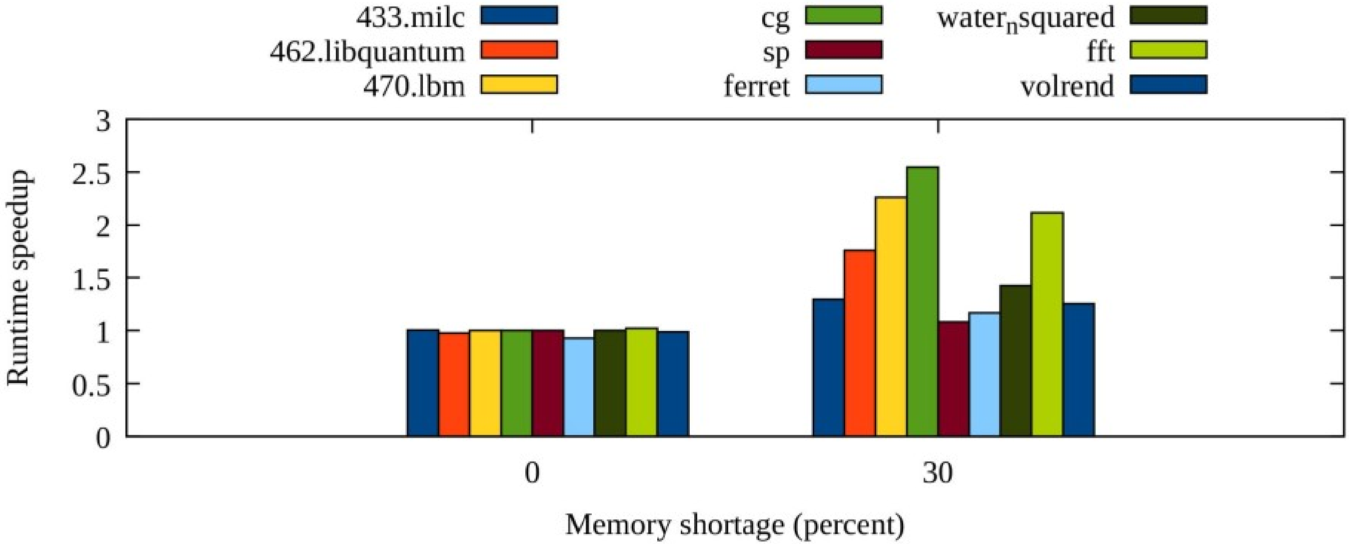
```
# Memory Footprints Distribution
percentile           0           25           50           75          100
       wss         0 B    9.520 MiB    9.543 MiB    9.785 MiB  107.039 MiB
       rss 104.820 MiB  104.820 MiB  104.820 MiB  104.820 MiB  104.820 MiB
       vsz 108.352 MiB  108.352 MiB  108.352 MiB  108.352 MiB  108.352 MiB
  sys_used   2.348 GiB    2.417 GiB    2.424 GiB    2.436 GiB    2.453 GiB

# Hotspot functions
# Samples: 589K of event 'cpu-clock:ppp'
# Event count (approx.): 147266750000
#
# Overhead  Command          Shared Object                         Symbol
# ........  ...............  ....................................  ...............................
#
    57.73%  swapper          [kernel.kallsyms]                     [k] pv_native_safe_halt
    40.26%  masim            masim                                 [.] do_seq_wo
     0.11%  python3          python3.11                            [.] _PyEval_EvalFrameDefault
     0.09%  ps               [kernel.kallsyms]                     [k] do_syscall_64
     0.05%  ps               [kernel.kallsyms]                     [k] memset_orig
     0.04%  ps               libc.so.6                             [.] open64
```

Utilizing DAMON for Access-aware Memory Management

- Profiling (e.g., GIF demo link)
  - For finding rooms to improve, e.g., capacity planning
- Profiling-guided Optimizations
  - Could be done on both offline and online
- Why not let kernel just (transparently) works?

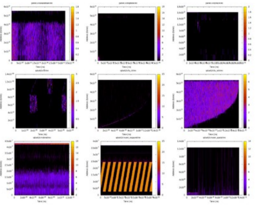


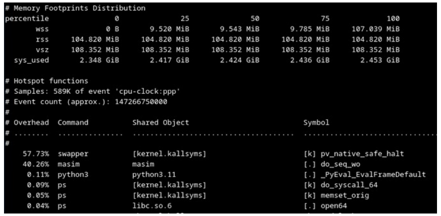
```
# Memory Footprints Distribution
percentile          0          25          50          75         100
       wss        0 B   9.520 MiB   9.543 MiB   9.785 MiB 107.039 MiB
       rss 104.820 MiB 104.820 MiB 104.820 MiB 104.820 MiB 104.820 MiB
       vsz 108.352 MiB 108.352 MiB 108.352 MiB 108.352 MiB 108.352 MiB
  sys_used   2.348 GiB   2.417 GiB   2.424 GiB   2.436 GiB   2.453 GiB

# Hotspot functions
# Samples: 589K of event 'cpu-clock:ppp'
# Event count (approx.): 147266750000
#
# Overhead  Command          Shared Object                         Symbol
# ........  ...............  ....................................  ..............................
#
    57.73%  swapper          [kernel.kallsyms]                     [k] pv_native_safe_halt
    40.26%  masim            masim                                 [.] do_seq_wo
     0.11%  python3          python3.11                            [.] _PyEval_EvalFrameDefault
     0.09%  ps               [kernel.kallsyms]                     [k] do_syscall_64
     0.05%  ps               [kernel.kallsyms]                     [k] memset_orig
     0.04%  ps               libc.so.6                             [.] open64
```



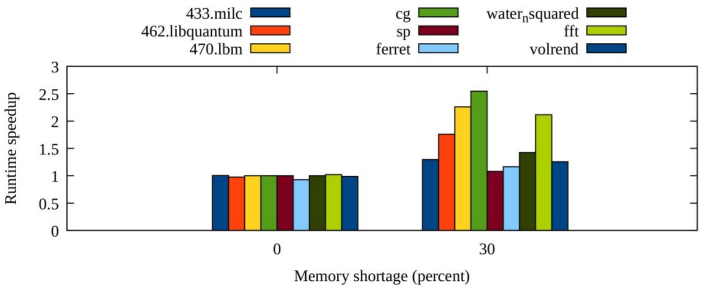
Utilizing DAMON for Access-aware Memory Management

- Profiling (e.g., GIF demo link)
  - For finding rooms to improve, e.g., capacity planning
- Profiling-guided Optimizations
  - Could be done on both offline and online
- Why not let kernel just (transparently) works?





```
# Memory Footprints Distribution
percentile            0           25           50           75          100
       wss          0 B    9.520 MiB    9.543 MiB    9.785 MiB  107.039 MiB
       rss  104.820 MiB  104.820 MiB  104.820 MiB  104.820 MiB  104.820 MiB
       vsz  108.352 MiB  108.352 MiB  108.352 MiB  108.352 MiB  108.352 MiB
  sys_used    2.348 GiB    2.417 GiB    2.424 GiB    2.436 GiB    2.453 GiB

# Hotspot functions
# Samples: 589K of event 'cpu-clock:ppp'
# Event count (approx.): 147266750000
#
# Overhead  Command          Shared Object                         Symbol
# ........  ...............  ....................................  ..............................
#
    57.73%  swapper          [kernel.kallsyms]                     [k] pv_native_safe_halt
    40.26%  masim            masim                                 [.] do_seq_wo
     0.11%  python3          python3.11                            [.] _PyEval_EvalFrameDefault
     0.09%  ps               [kernel.kallsyms]                     [k] do_syscall_64
     0.05%  ps               [kernel.kallsyms]                     [k] memset_orig
     0.04%  ps               libc.so.6                             [.] open64
```



DAMOS: Data Access Monitoring-based Operation Schemes

- The other face of DAMON
- Let users define schemes
  - Memory operation actions to apply to regions of specific access pattern
- Once per user-defined time interval
  - find the regions of the condition from the snapshot and apply the action
- Finding optimum “access pattern” on dynamic environments is challenging
  - Uncontrolled DAMOS could byte you!

```
# # pageout memory regions that not accessed for >=5 seconds
# damo start --damos_action pageout --damos_access_rate 0% 0% --damos_age 5s max
```

Use Cases

Proactive Cold Memory Reclamation

- Proactively find cold pages and reclaim
- Reduce memory footprint without performance degradation
- Reduce memory pressure occurrences and help (automated) capacity planning
- AWS Aurora Serverless v2 uses this for memory auto-scaling

Memory Tiering

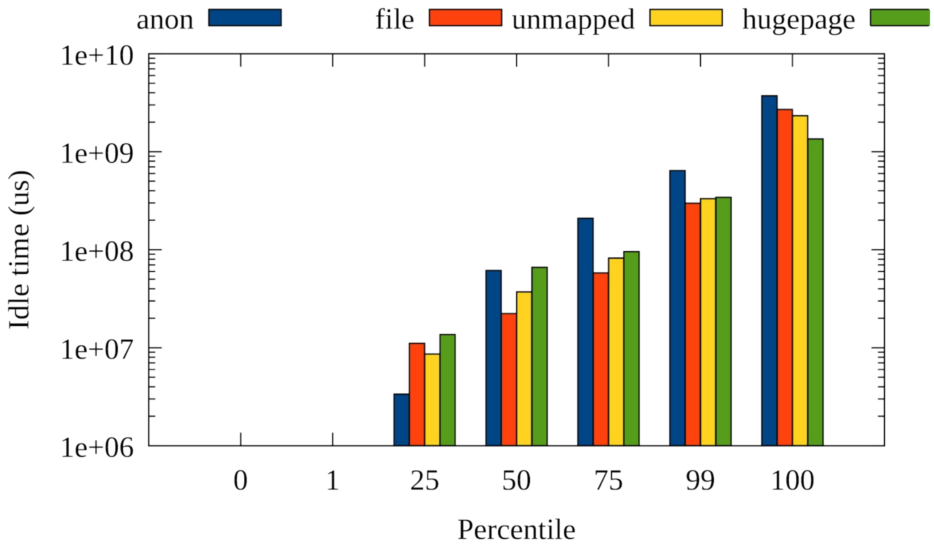
- Migrate hot data in slower NUMA nodes to faster nodes, cold data in opposite direction
  - e.g., CXL and DRAM nodes
- SK Hynix developed and utilizing this for their Heterogeneous Memory SDK
- Meta's self-tuned version shows ~7.34% performance improvement on a test workload (Taobench)
- Cgroup fairness-aware extension is also available in RFC

Access-aware Dynamic Memory Interleaving

- Memory interleaving: interleave placement on allocation time for bandwidth control
- Dynamically interleave (migrate) data for dynamic access pattern in access-aware order
- Micron developed for their internal project
- Micron’s test shows 25% performance improvement

Page Level Data Access Monitoring

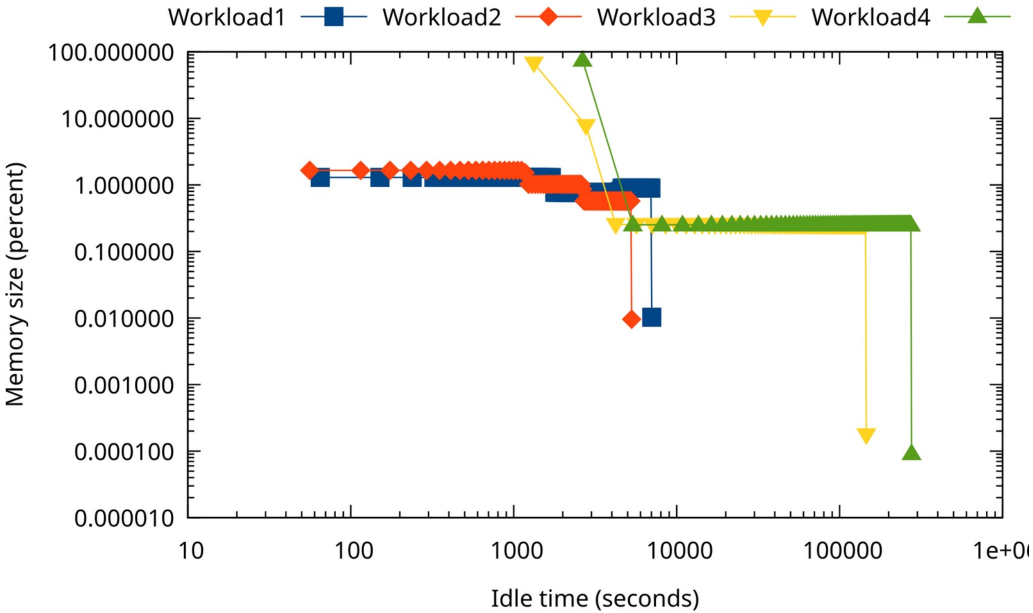
- Different types of pages have different management mechanisms
- Breakdown data access pattern for specific types of pages
- Developed by Meta for hugepage and LRU-active pages access pattern profiling

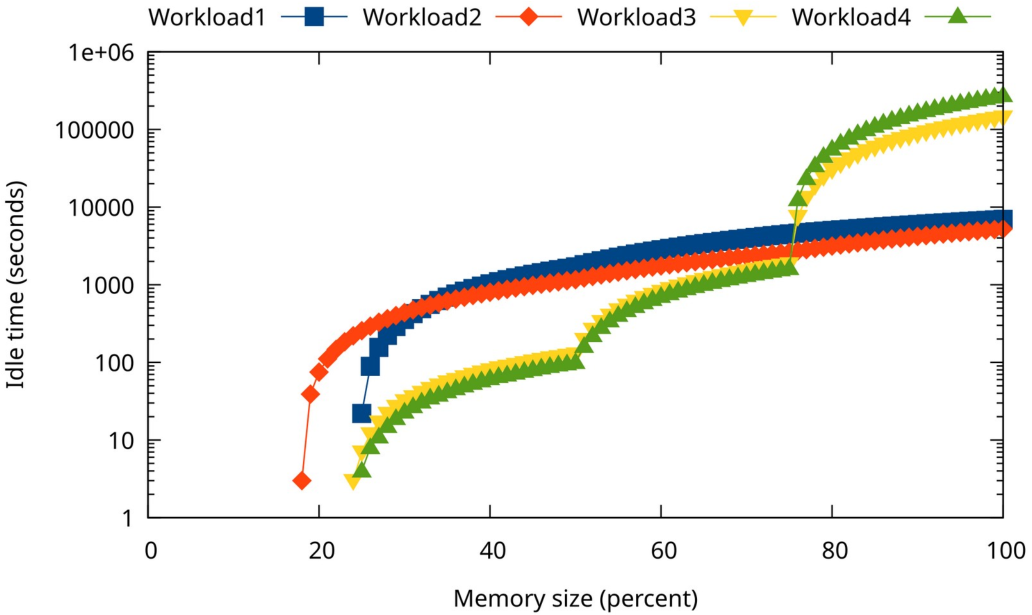


Per-page type access pattern of a production workload

Fleet-wide Data Access Monitoring

- Transform DAMON snapshot into memory idle time distribution (percentiles)
- Can easily aggregated and intuitively visualized (idle time percentiles or cold memory tails)
- Developed by Meta for fleet-wide real workloads access pattern profiling





Cold memory tails (left) and idle memory time percentiles (right) of real workloads

Getting Started

Where to Get Started: https://damonitor.github.io

- Project website
- Contains getting started guides and all resources for users and developers
- Should have all you need to get started
  - If not, report it please

Availability

- Merged into the mainline from v5.15
- Backported and enabled on major Linux distro kernels
  - Major distros: Alma, Amazon, Android, Arch, CentOS, Debian, Fedora, Oracle, …
- DAMON user-space tool (damo) is available on major packaging systems
  - Arch, Debian, Fedora, PyPi, ...

Interfaces

- DAMON user-space tool: Recommended for general usages from user-space
- DAMON modules: Recommended for specific usages
- DAMON sysfs interface: Recommended for user-space program development
- Kernel API: Recommended for kernel programmers

Community: For Questions, Help, Patch Reviews

- Public mailing list (https://lore.kernel.org/damon)
- Bi-weekly virtual meetup
  - Occasional/regular private meetings on demand
- Not used to mail-based development? Try hkml
  - Developed and maintained for DAMON and Linux kernel developers
- The future of DAMON is open and up to you
  - “Prefer random evolution over intelligent design”

Summary: That’s DAMON

- DAMON is a Linux kernel subsystem
  - For practical access monitoring based holistic and observable memory managements
- Companies, researchers and individuals are using it for better memory managements
- The future is open and up to the community
  - Make your selfish voice

Questions?

- You can also ask questions anytime to
  - Maintainer: sj@kernel.org
  - Public mailing list (https://lore.kernel.org/damon)
  - Bi-weekly virtual meetup
  - Occasional/regular private meetings on demand
  - Project website (https://damonitor.github.io)

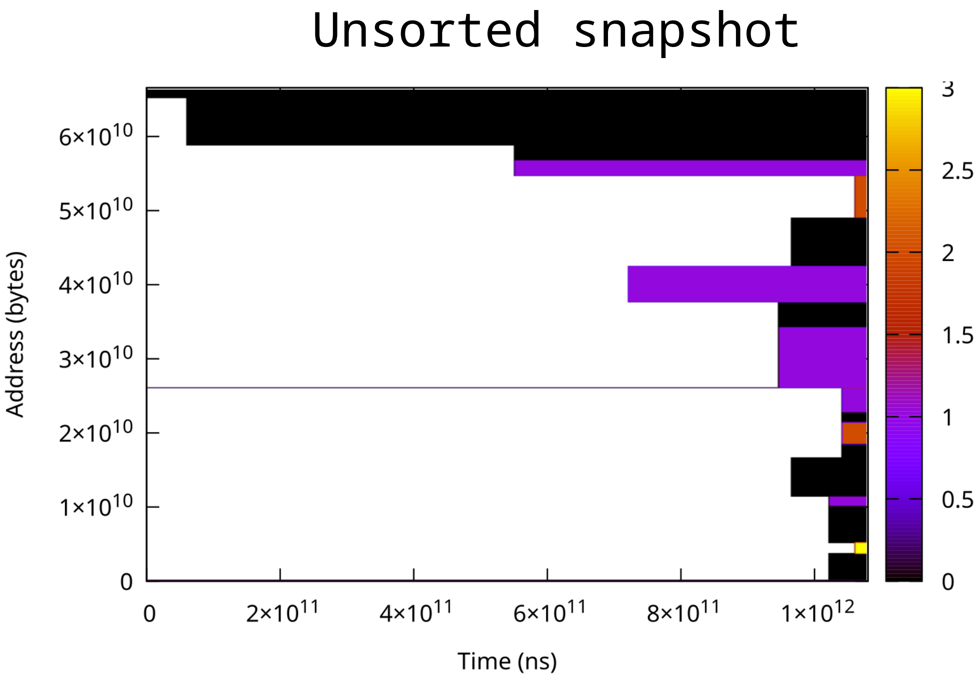
Backup Slides

DAMON_STAT: Recommended Way For System-wide Access Monitoring

- Kernel module running DAMON for the entire physical address space
- Use intervals auto-tuning with the suggested auto-tune parameters
- Extract Idle time percentile
  - distribution of per-byte memory idle times (time the byte was not accessed)
  - P75 idle time 2minutes: 75 percent of the memory was accessed at least once in last 2 minutes; rest 25 percent of memory was not accessed at all for last 2 minutes
- Extract estimated memory bandwidth
  - Memory bandwidth estimated based on access events that captured in the last snapshot
- Recommended way for system-wide access monitoring
  - Easy to enable (CONFIG_DAMON_STAT_DEFAULT_ENABLED=y), aggregate, compare
  - Can be enabled/disabled at build, boot time and runtime

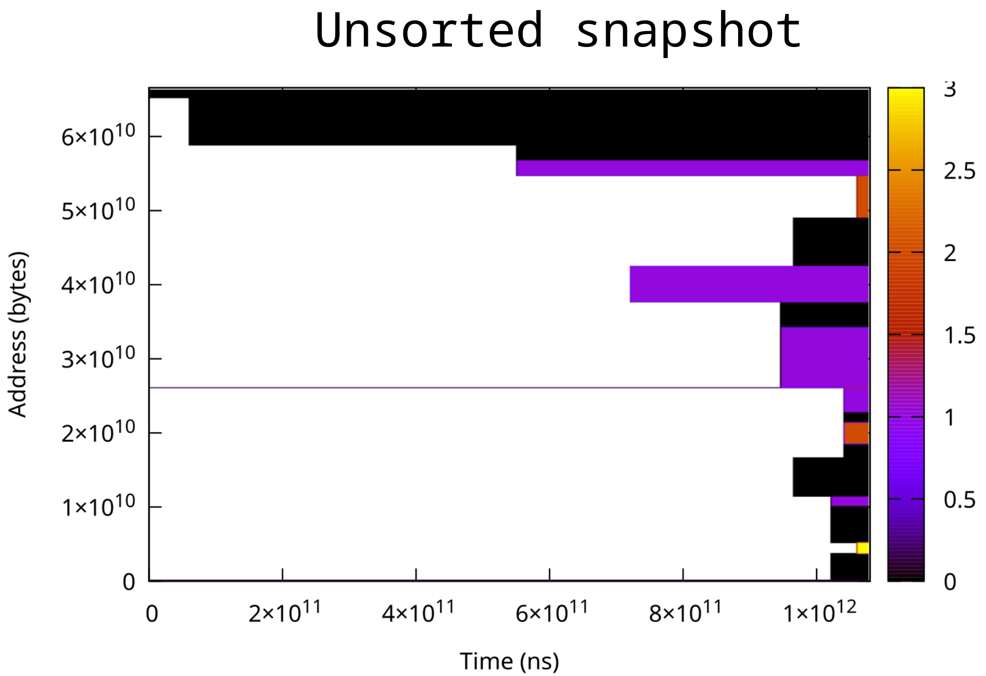
Idle Time Percentiles

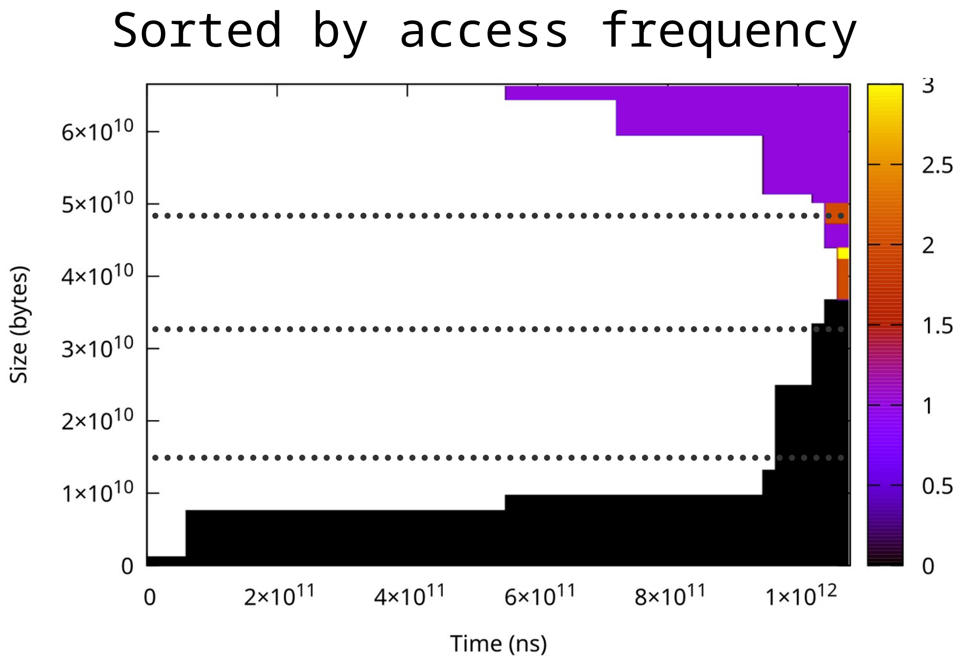
- Idle time: How long the region kept being not accessed (access frequency 0)
- Idle time percentiles: Percentiles of sorted per-byte idle times



Idle Time Percentiles

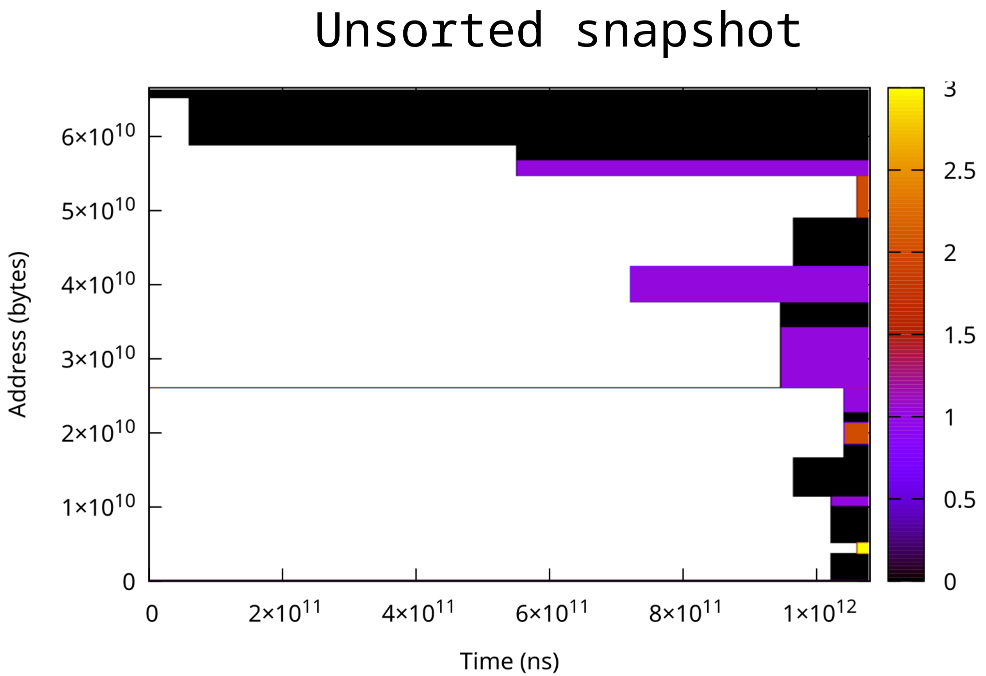
- Idle time: How long the region kept being not accessed (access frequency 0)
- Idle time percentiles: Percentiles of sorted per-byte idle times

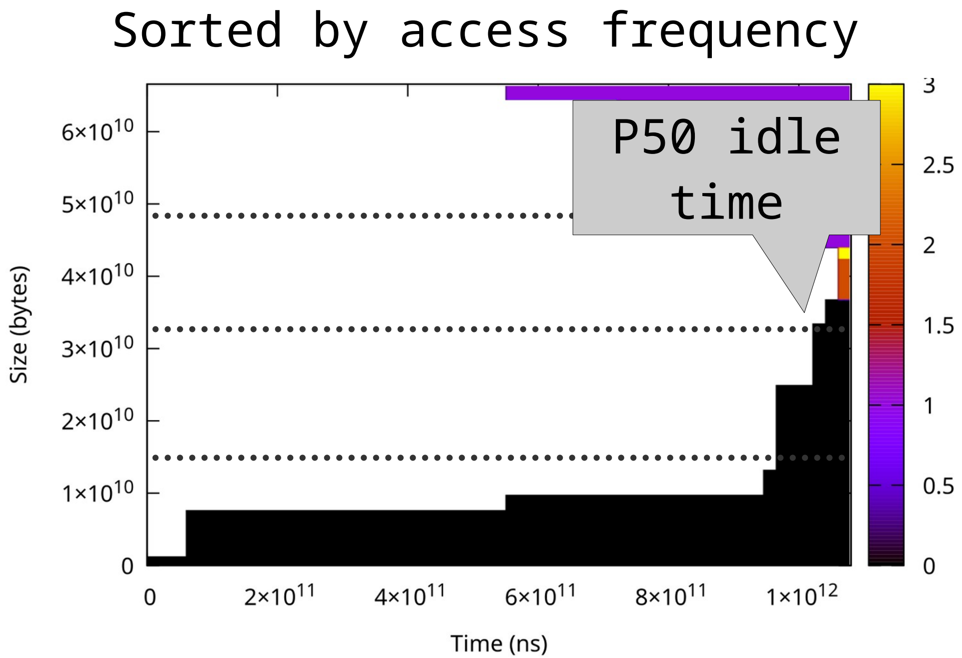




Idle Time Percentiles

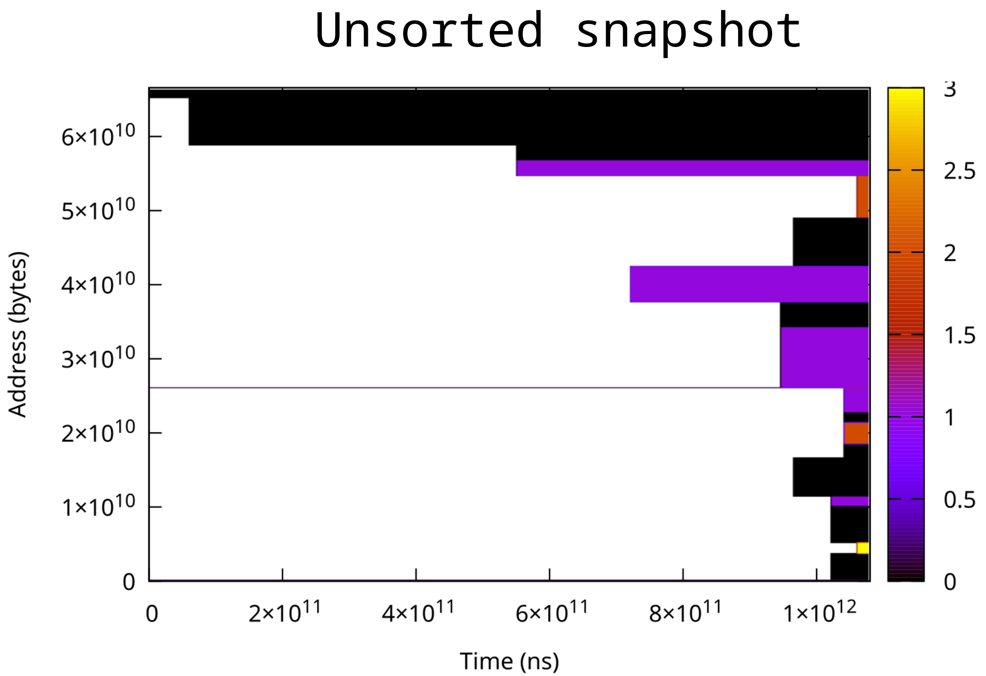
- Idle time: How long the region kept being not accessed (access frequency 0)
- Idle time percentiles: Percentiles of sorted per-byte idle times

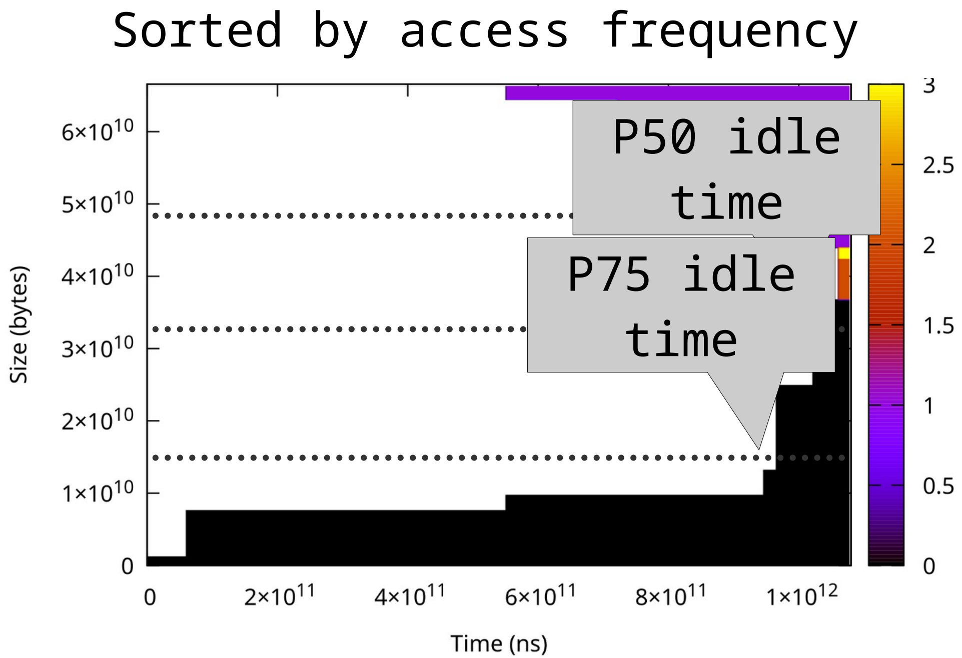




Idle Time Percentiles

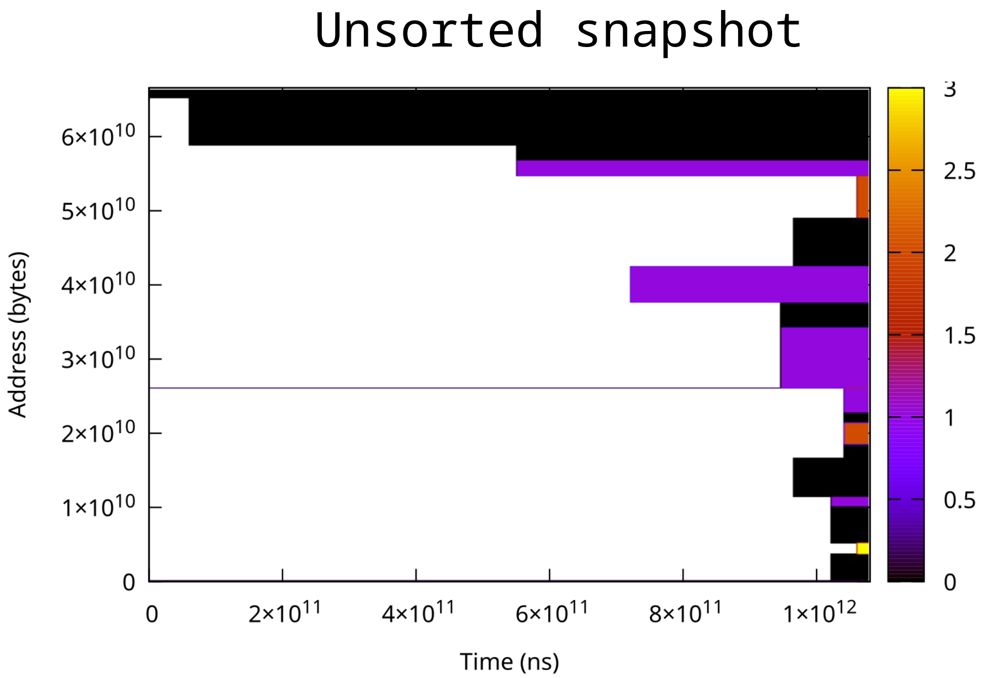
- Idle time: How long the region kept being not accessed (access frequency 0)
- Idle time percentiles: Percentiles of sorted per-byte idle times

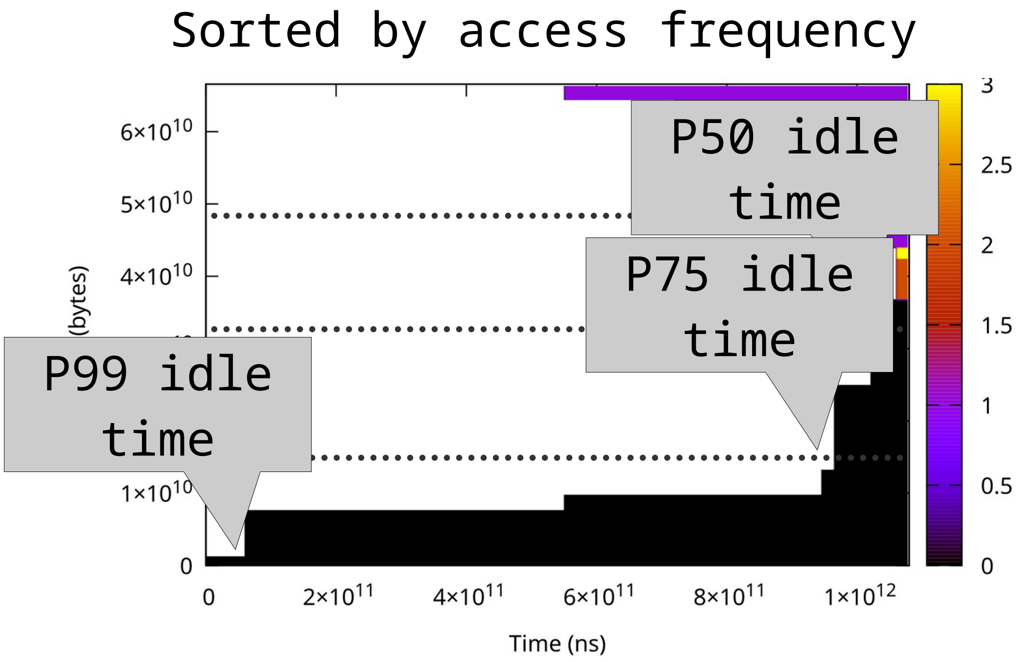




Idle Time Percentiles

- Idle time: How long the region kept being not accessed (access frequency 0)
- Idle time percentiles: Percentiles of sorted per-byte idle times



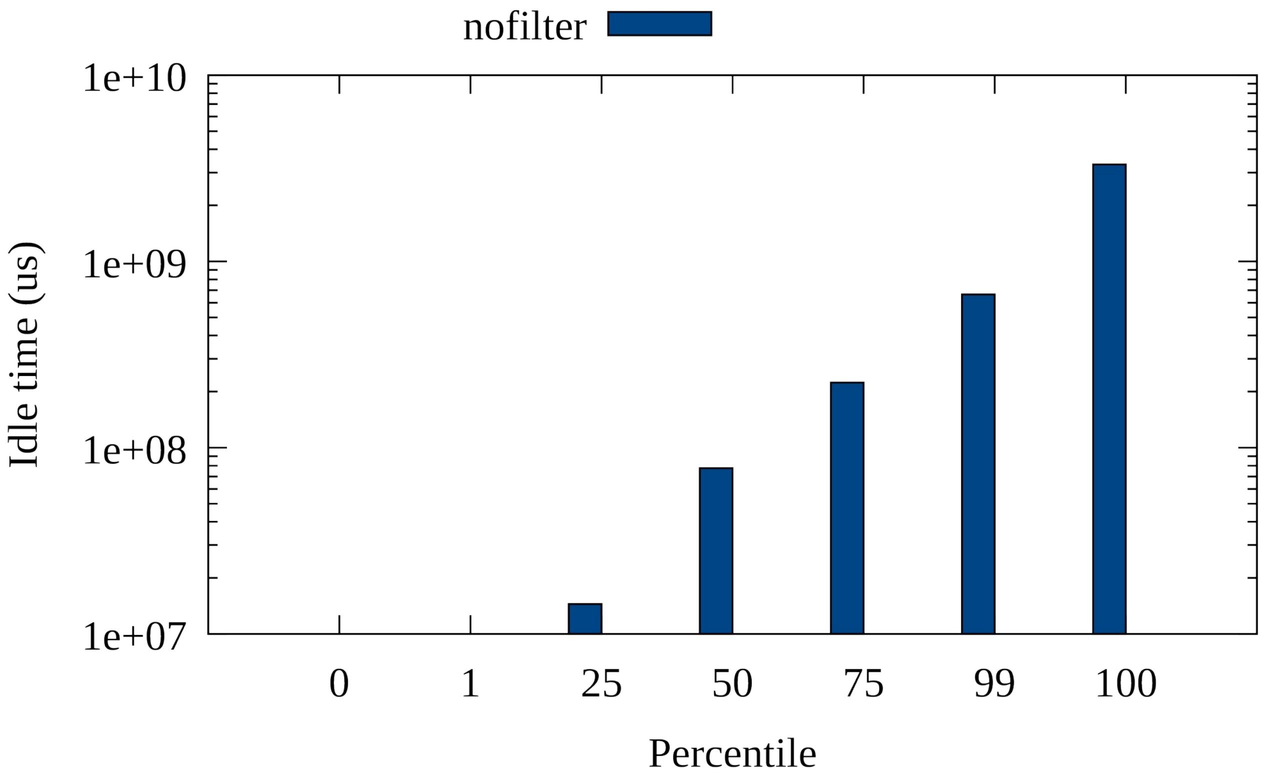


DAMON_STAT: Recommended Way For System-wide Access Monitoring

- Kernel module running DAMON for the entire physical address space
- Use intervals auto-tuning with the suggested auto-tune parameters
- Extract Idle time percentile
  - distribution of per-byte memory idle times (time the byte was not accessed)
  - P75 idle time 2minutes: 75 percent of the memory was accessed at least once in last 2 minutes; rest 25 percent of memory was not accessed at all for last 2 minutes
- Extract estimated memory bandwidth
  - Memory bandwidth estimated based on access events that captured in the last snapshot
- Recommended way for system-wide access monitoring
  - Easy to enable (CONFIG_DAMON_STAT_DEFAULT_ENABLED=y), aggregate, compare
  - Can be enabled/disabled at build, boot time and runtime

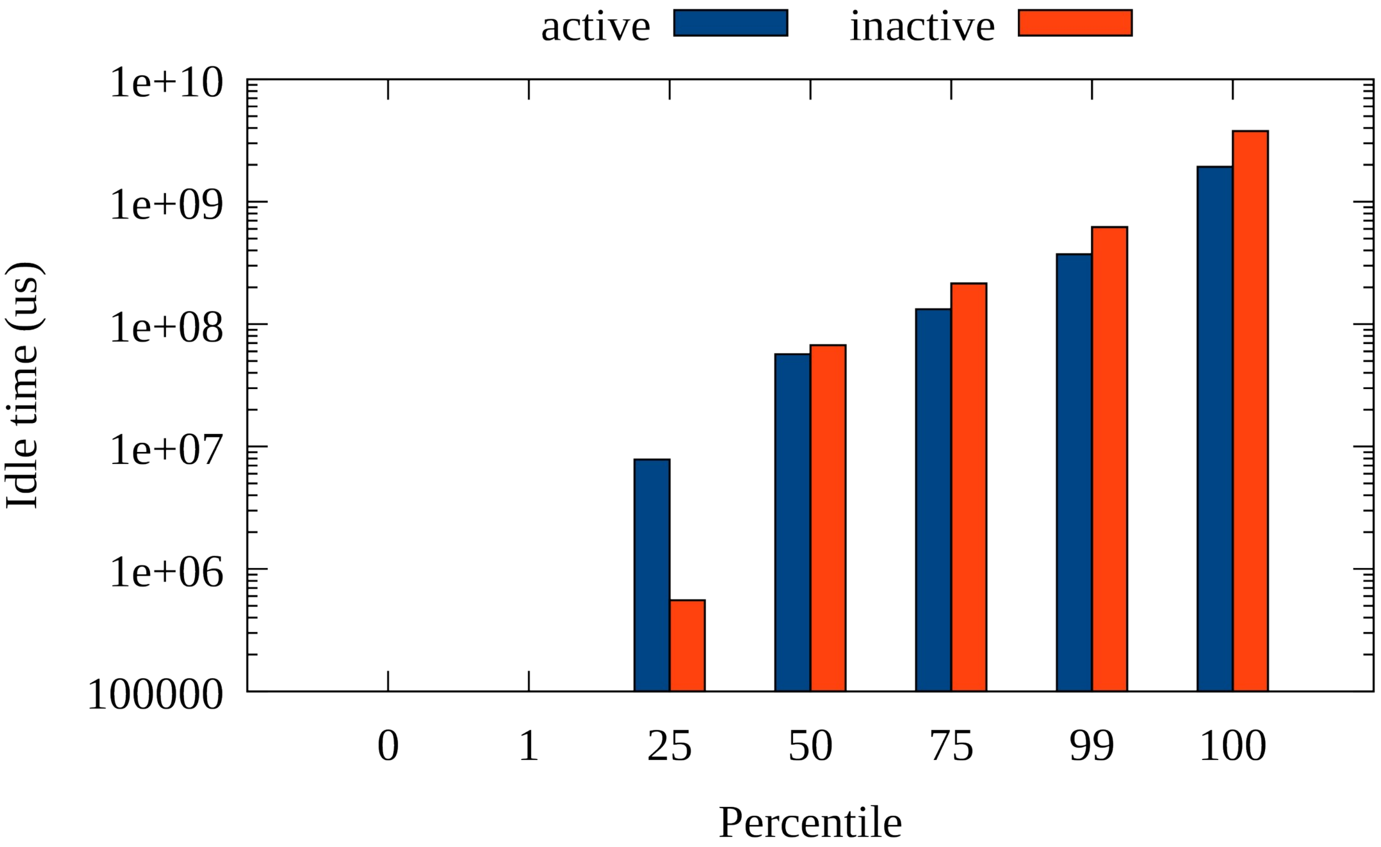
Results on a Real Workload: Auto-tuned Total Memory Idle Time Percentiles

- Small hot memory, exponentially increasing idle time (long tail of cold pages)



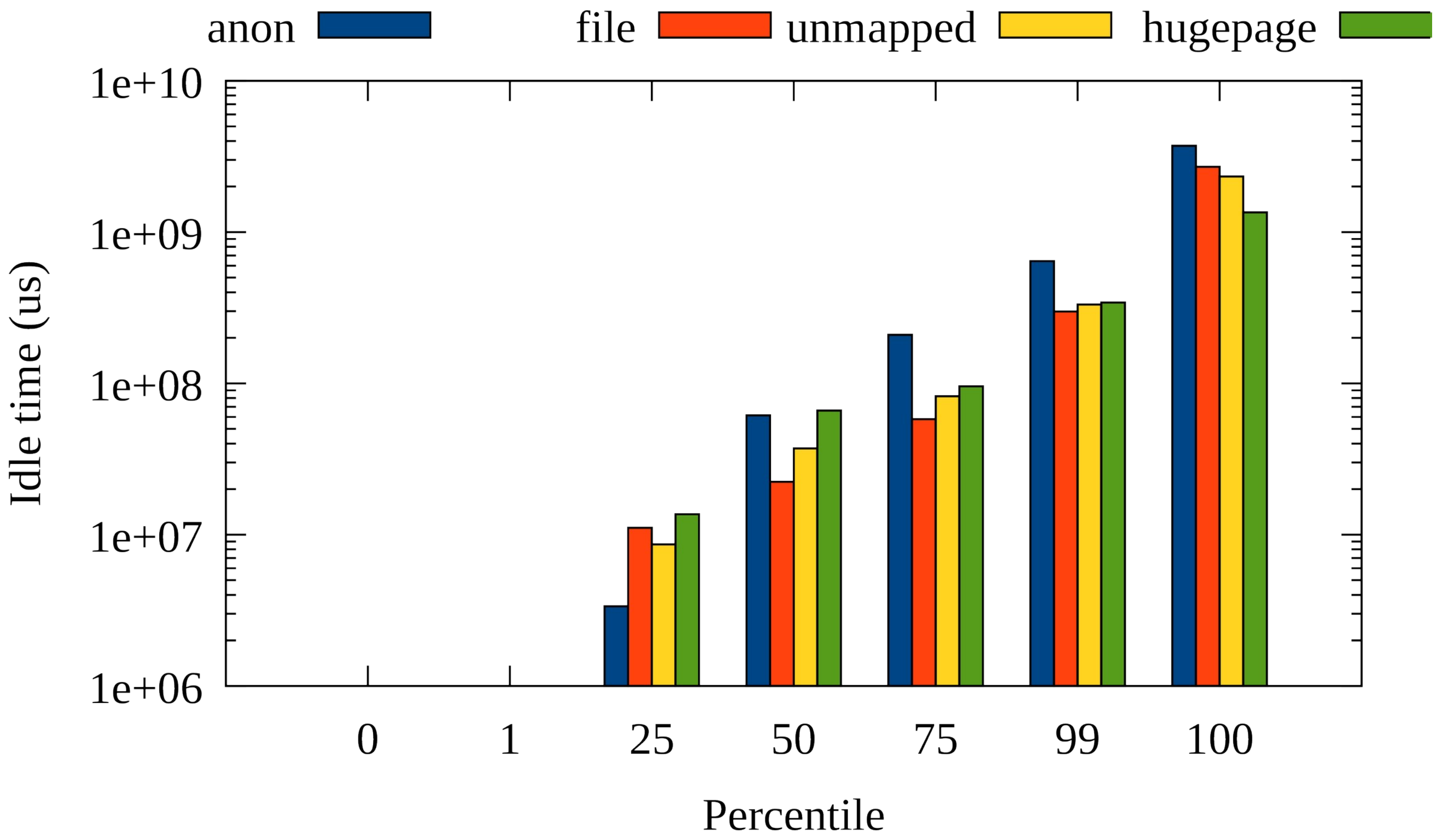
Results on a Real Workload: Active vs Inactive Pages Idle Time Breakdown

- Active pages have rooms to be more hot than inactive
  (ideally, p100 of active should < p0 of inactive)



Results on a Real Workload: Per Page Type Idle Time Breakdown

- You can check if your workload has expected access pattern



DAMOS Filter: Fine-Control Access-aware System Operation Targets

- Define target memory with non-access-pattern information
  - Page level filters: anon, owned cgroup, hugepage, LRU-activeness
  - Non-page level filters: address
  - “pageout cold pages *of NUMA node 1 that associated with cgroup A and file-backed”*
  - Can be useful for fine-grained monitoring, too
    - (“stat”, instead of “pageout”)

DAMOS Quota: Control Access-aware System Operation Aggressiveness

- Six fixed thresholds (min/max size, access frequency, age) are unnecessary in many cases
- Setting thresholds flexibly and controlling aggressiveness works in many cases
  - Single control knob
- Quota set the aggressiveness limit as amount of memory to apply action per a time interval
- Access pattern based prioritization is applied under the quota
- “pageout cold pages *up to 100 MiB per second using <2% CPU time, coldest ones first*”

Quota Auto-tuning: Auto-tuned Access-aware System Operations

- Quota tuning is manual and repetitive
- Change the question for user: How to do (mechanism) → What to achieve (final goal)
- Let users specify goal of the quota as a value of a metrics
  - Metrics: PSI level, NUMA node memory utilization, workload’s latency, bandwidth, TPS, …
  - e.g., “reclaim cold pages aiming 0.5% memory PSI”
- DAMOS adjusts quota using feedback loop, for current value of the metric
  - e.g., If memory PSI is 0.1% increase quota for reclaiming cold pages (reclaim more warm pages)

Controlled and Auto-tuned Access-aware System Operation Performance

- Parsec3/splash2x.fft
- Page out regions that not accessed for >=5 seconds, up to 1GiB/sec, using up to 100ms/sec, aiming 10ms/sec memory pressure stall

	Runtime	RSS
Baseline	50.489s	10.005 GiB
+DAMOS-reclaim	120s	4.955 GiB
+Quota	51.772s	8.527 GiB
+Goal	49.741s	9.721 GiB

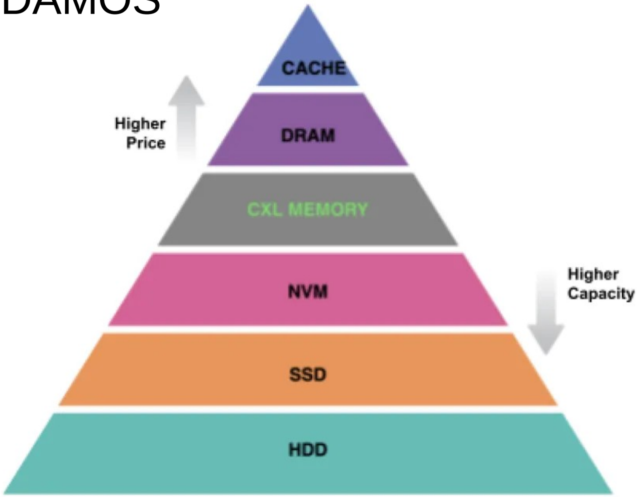
Real-world DAMON Use Cases: Proactive Reclamation and CXL Memory Tiering

Proactive Reclamation

- Reactive reclamation: Reclaim cold memory when memory pressure happens
- Proactively reclamation: Reclaim cold memory before memory pressure
- Benefit 1: Reduce memory footprint without performance degradation
- Benefit 2: Minimize degradation from direct reclamation
- Known usages: Google, Meta, and Amazon
  - Each company uses its own implementation for its usage
- AWS uses DAMOS-based implementation since 2022

CXL Memory Tiering

- CXL-tiered memory: Put CXL memory between DRAM and NVM
  - Pros: Higher capacity with lower price (higher efficiency)
- Challenge: Dynamic placement of pages (CXL mem is slower than DRAM)
- DAMON-based approach: Place hot pages on DRAM node, Place cold pages on CXL node
- SK hynix developed their CXL memory SDK (HMSDK) using DAMOS
  - Reports ~12.9% speed up

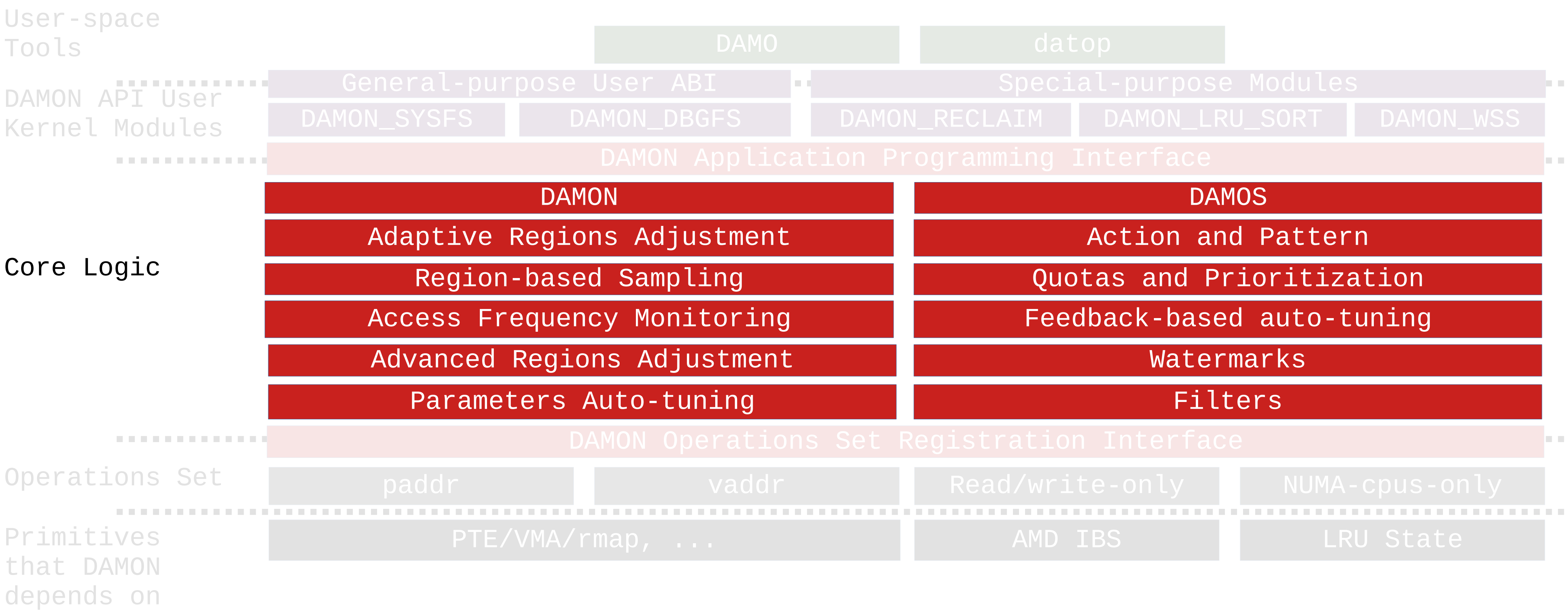


Architectures

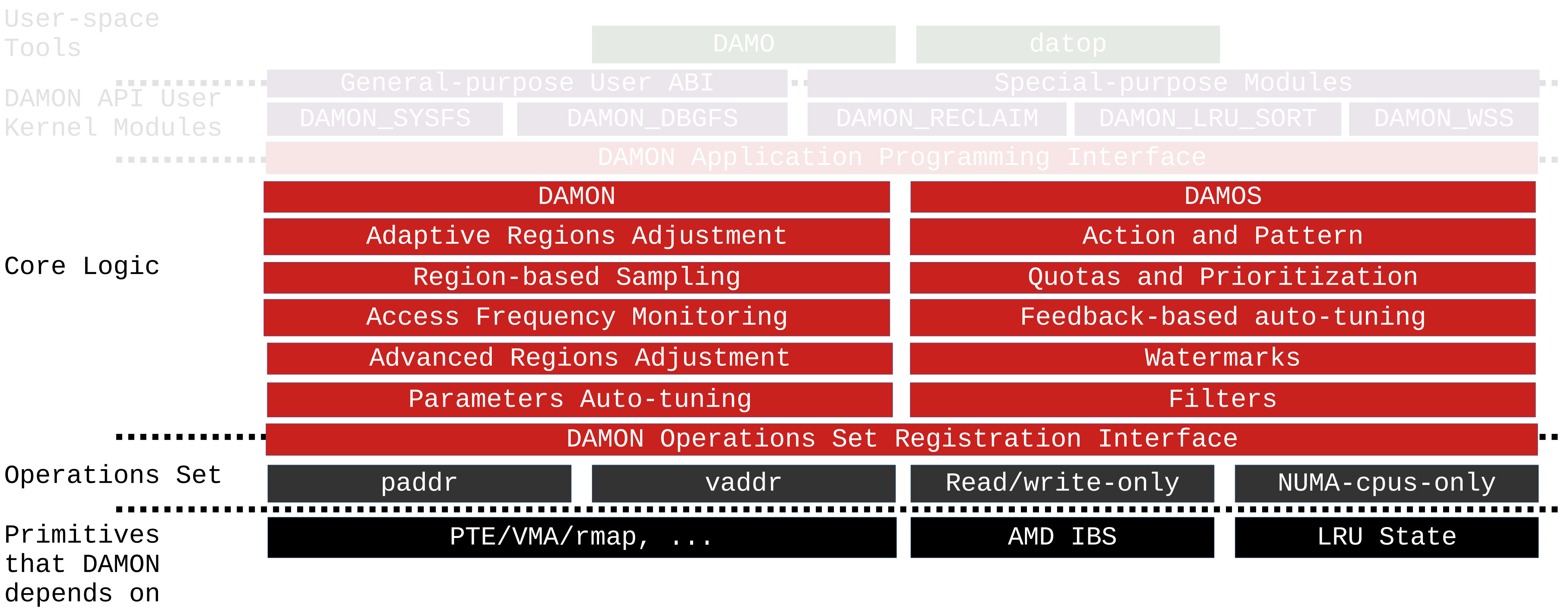
Execution Model: Kernel Thread per Requests

- “struct damon_ctx”: Data structure for DAMON user input/output containing
  - User requests: target address space, address range, intervals, DAMOS schemes
  - Operation results: access snapshot, DAMOS stats
- “kdamond”: DAMON worker thread
  - Create one kdamond per “damon_ctx”
    - In future, could support multiple “damon_ctx” per kdamond
    - In future, could separate DAMOS to another thread (maybe useful for cgroup charging)
  - Allows async DAMON execution and multiple kdamonds (CPUs) scaling

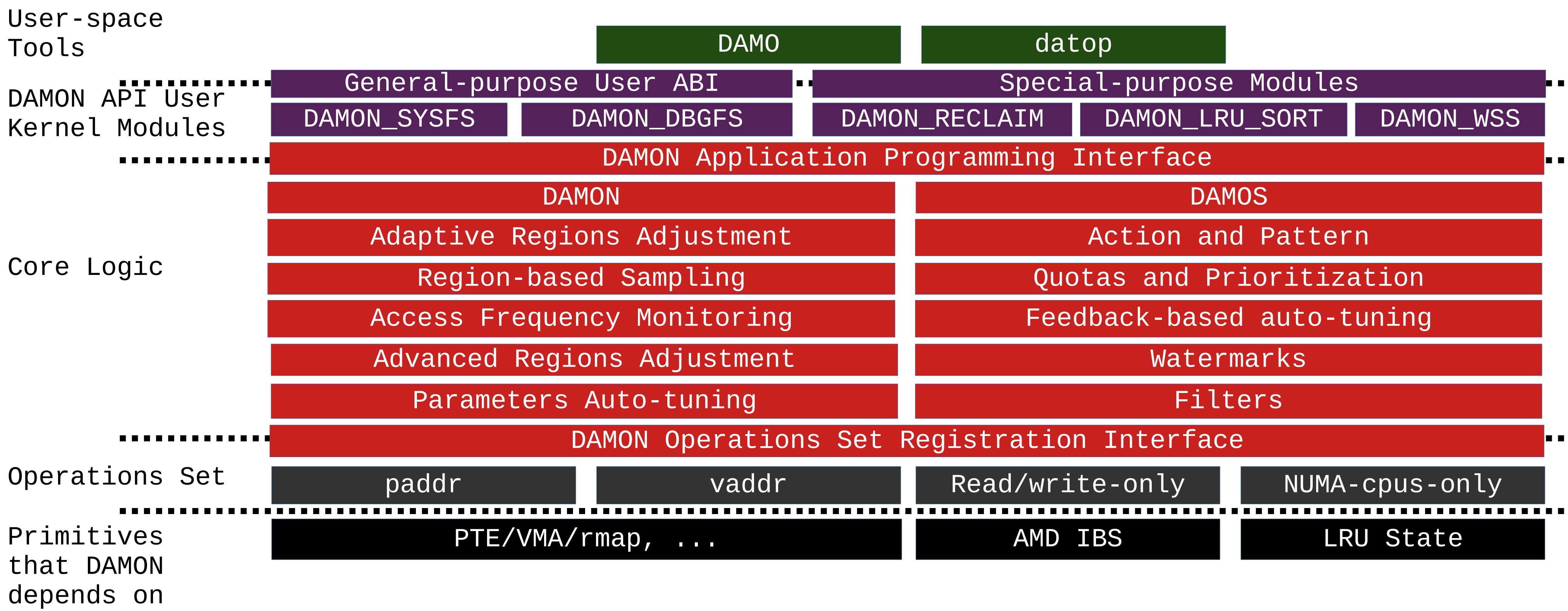
Extensible Layers



Extensible Layers



Extensible Layers



DAMOS Quotas: Intuitive Aggressiveness Control

- Before applying DAMOS schemes
  - Set temperature-based priority score of each region
  - Build “priority score”: “total size of regions of the priority” histogram
  - Find lowest priority threshold for the scheme meeting the quota
  - Skip applying action to regions having lower-than-threshold priority scores
- Single snapshot and histogram iteration: O(<=user-defined-N)
- Quota auto-tuning: A simple proportional feedback algorithm
  - Reward metrics: Arbitrary user-input or self-retrievable metrics like memory PSI

DAMOS Filters: Fine-grained Target Selection

- Before applying DAMOS action, check the properties of region and skip action if needed
- Non-page granular (high level) filters
  - Filtered out before applying actions
  - Address ranges (e.g., NUMA nodes or Zone)
  - DAMON-defined monitoring target (e.g., process)
- Page granular (low level) filters
  - Filtered out in the middle of actions in page level
  - Anon/File-backed
  - Belonging memory cgroup
  - page_idle()

Pseudo-code of DAMON v5.15

```
While True:
    for region in regions:
        if region.accessed():
            region.nr_accesses += 1
    sleep(sampling_interval)
    if now() % aggregation_interval:
        merge(regions)
        user_callback(regions)
        for region in regions:
            region.nr_accesses = 0
        split(regions)
```

DAMON accuracy on Low-locality Space/Workloads

- It is proven to work on real world products for years
- Pareto principle and unconcious bias will make the pattern
  - Entropy-full situation is when the data center is doom-ed
- “age” avoid immature decision
- More works for accuracy improvement will be continued
- DAMON could be decoupled with the region-based mechanisms in future
- Let’s collect data and continue discussions together

Can DAMON Extended for Non-snapshot Access Patterns?

- TL; DR: Yes, why not?
- DAMON is for any access information; Snapshot is one of the representations
- If the information/representation is useful for users, DAMON can add support
- We started discussion for Memory bandwidth visibility

Can DAMON Use features Other than PTE Accessed bits?

- The extensible layer allows it
- AMD IBS and page fault-based appaoraches (e.g., PTE_NONE) are on the table
- In future, if GPU provides access check feature, we can extend to use it
- Such extension would allow
  - More lightweight and precise monitoring
  - Access source, read/write-aware monitroing
  - Kernel memory access monitoring

DAMOS for Efficient and Fine-grained Data Access Monitoring

- DAMOS_STAT
  - Special action making no system change but expose the scheme-internal information
  - Let user knows which of the memory are eligible for the scheme
- With DAMOS filters, can do page level properties-based monitoring
  - “How much of >2 minutes unaccessed memory are in hugepages and belong to cgroup A?”
- With DAMOS quotas, can do overhead-controlled monitoring