Page-level and Fleet-wide Data Access Monitoring for Meta

SeongJae Park (SJ) <sj@kernel.org> <sjpark@crusoe.ai>

東京 2025 LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DEC. 11-13, 2025

Device and Specific Purpose Memory MC

About Speaker

- Name: SJ
- Maintain a kernel subsystem, DAMON
- Working for crusoe.ai (We’re hiring!)
  - Was Meta worker when this talk is submitted
  - Everything is open source and upstreamed
  - All opinions are speaker’s own

Table of Contents

- Why access pattern observability (1 mins)
- DAMON in Nutshell (4 mins)
- Page Level Monitoring (13 mins)
- Fleet-wide monitoring (17 mins)
- QnA (10 mins)

TL; DR

- Set `DAMON_STAT_ENABLED_DEFAULT=y`
- And be happy observer :)

Why Meta (and You) Need Access Pattern Observability

We Love Data, Hate Memory

- Data: Our precious (money generator)
  - Source of $$
- Memory: The Place for Data
  - Sinkhole of $$
- Diverse types of data and memory exist
- Efficient memory management saves $$

Access Pattern for Efficient MM

- Efficient MM
  - More Precious Data in Less Memory
- Data Access Pattern
  - To make action on [non-]precious data
  - To self-aware and assess
  - To repeat

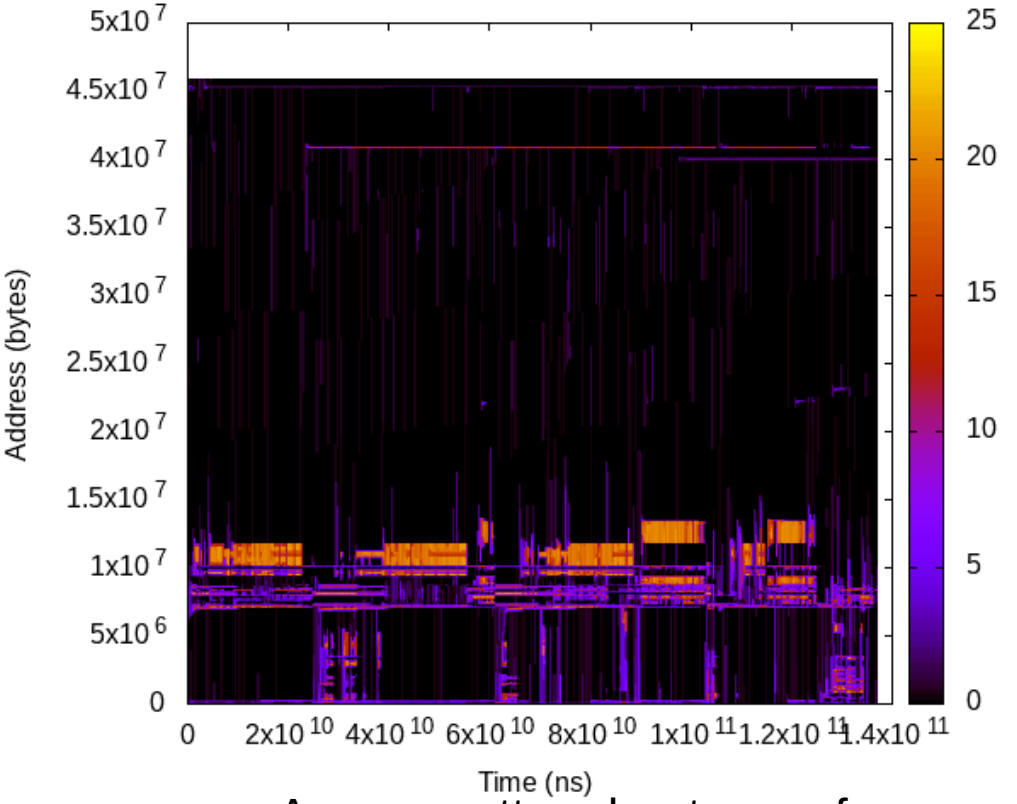
DAMON in Nutshell

What It Does

- Spawn a kernel thread that
- See if each page is accessed, every 5ms
- Inform users the findings, every 100ms
- Time intervals can be [auto-]tuned

Providing Access Information

- Location
- Frequency
- Stability
- Recency



Access pattern heatmap of Splash2x/Raytrace

Lightweight and Accurate

- Utilizes adaptive sampling/aggregation
- The upper-limit overhead is tunable
  - Regardless of the memory size (scalable)
- 0.0x% single CPU use on real workloads
  - It’s up to user; multiple CPUs use is also ok

Availability and Usages

- Available on >=5.15 upstream kernels
- Enabled on most distro kernels
- Being used by products and researches

DAMOS: Second Face of DAMON

- DAMon-based Operation Schemes
  - “Page out cold memory”
  - “Use huge pages for hot memory”
- Turn DAMON into access-aware memory management system operations engine

DAMON in One Sentence

“DAMON is a Linux kernel subsystem for efficient access monitoring and access-aware system operations.”

Community

- Waiting for any **selfish** discussion
- Public channels
  - Mailing list: damon@lists.linux.dev
  - Project website: https://damonitor.github.io/
- Private channels
  - Maintainer email: sj@kernel.org
  - DAMON Beer/Coffee/Tea Meetup

Page Level Monitoring

Motivation: Huge Pages Hotness

- Meta knows huge pages matter

```
$ ./damo/damo report access
heatmap:
655555555555555555555544444444544456666333333455555558999998100000002444446631
# min/max temperatures: -92,557,696,000, 60,740,989,500, column size: 793.600 MiB
intervals: sample 723.107 ms aggr 14.462 s (max access hz 1.383)
0   addr 4.000 GiB    size 20.000 MiB  access 0 hz     age 2 m 53.546 s
1   addr 4.020 GiB    size 2.000 MiB   access 1.383 hz age 3 h 4 m 37.999 s
2   addr 4.021 GiB    size 6.046 GiB   access 0.069 hz age 1 m 26.773 s
[...]
18  addr 64.780 GiB   size 1.220 GiB   access 0 hz     age 10 m 7.410 s
memory bw estimate: 3.889 GiB per second
total size: 62.000 GiB
record DAMON intervals: sample 723.107 ms, aggr 14.462 s
# target 4 % accesses per 3 aggrs, [5 ms, 10 s] sampling interval
```

Motivation: Huge Pages Hotness

- Meta knows huge pages matter

```
$ ./damo/damo report access
heatmap:
655555555555555555555544444444544                    5589999981000000024444466 31
# min/max temperatures: -92,557,690,                    00, column size: 793.600 MiB
intervals: sample 723.107 ms aggr 14.                 s (max access hz 1.383)
0   addr 4.000 GiB    size 20.000 MiB   access 0 hz     age 2 m 53.546 s
1   addr 4.020 GiB    size 2.000 MiB    access 1.383 hz age 3 h 4 m 37.999 s
2   addr 4.021 GiB    size 6.046 GiB    access 0.069 hz age 1 m 26.773 s
[...]
18  addr 64.780 GiB   size 1.220 GiB    access 0 hz     age 10 m 7.410 s
memory bw estimate: 3.889 GiB per second
total size: 62.000 GiB
record DAMON intervals: sample 723.107 ms, aggr 14.462 s
# target 4 % accesses per 3 aggrs, [5 ms, 10 s] sampling interval
```

Is this a huge page?

Avail but Uncollected Info

- DAMON lacks the info
  - works on regions, not pages
- DAMOS has the info
  - Works on pages
  - Supports page type-based action filtering

Page Level Monitoring Support

- DAMOS provides the info to DAMON
- DAMON exposes it to users
- User-space tool
  - runs DAMOS with STAT action and filter
  - Visualize the DAMON-exposed info

Page Level Monitoring Output

- Df-passed: DAMOS filter passed bytes

```
$ sudo damo report access –damos_filter allow hugepage_size 2M max
[...]
0   addr 4.000 GiB    size 8.000 MiB   access 0.069 hz age 2 m 53.546 s     df-passed 0 B
1   addr 4.008 GiB    size 12.000 MiB  access 0 hz   age 2 m 53.546 s       df-passed 0 B
2   addr 4.020 GiB    size 816.000 KiB access 1.383 hz age 3 h 4 m 37.999 s df-passed 2.000 MiB
[...]
11  addr 25.053 GiB   size 3.023 GiB   access 0 hz   age 3 m 8.008 s        df-passed 20.000 MiB
12  addr 28.075 GiB   size 123.578 MiB access 0.069 hz age 3 m 8.008 s      df-passed 0 B
13  addr 28.196 GiB   size 185.367 MiB access 0.069 hz age 3 m 8.008 s      df-passed 2.000 MiB
[...]
24  addr 64.780 GiB   size 1.220 GiB   access 0 hz   age 10 m 7.410 s       df-passed 0 B
memory bw estimate: 3.079 GiB per second  df-passed: 10.787 MiB per second
total size: 62.000 GiB  df-passed 86.000 MiB
```

Page Level Monitoring Output

- Df-passed: DAMOS filter passed bytes

```
$ sudo damo report access –damos_filter allow hugepage_size 2M max
[...]
0   addr 4.000 GiB    size 8.000 MiB   access 0.069 hz age 2 m 53.546 s     df-passed 0 B
1   addr 4.008 GiB    size 12.000 MiB  access 0 hz   age 2 m 53.546 s       df-passed 0 B
2   addr 4.020 GiB    size 816.000 KiB access 1.383 hz age 3 h 4 m 37.999 s df-passed 2.000 MiB
[...]
11  addr 25.053 GiB   size 3.023 GiB   ccess 0 hz   age 3 m 8.008 s        df-passed 20.000 MiB
12  addr 28.075 GiB   size 123.57[illegible]  cess 0.069 hz age 3 m 8.008 s      df-passed 0 B
13  addr 28.196 GiB   size [illegible]                                  8
[...]
24  addr 64.780 GiB   size [illegible]                                  7.
memory bw estimate: 3.079 GiB per second  df-passed: 10.787 MiB
total size: 62.000 GiB  df-passed 86.000 MiB
```

This is quite consistently being accessed

And it is using a 2MiB-size huge page (we are doing ... GOOD?)

Page Level Monitoring Output

- Df-passed: DAMOS filter passed bytes

```
$ sudo damo report access –damos_filter allow hugepage_size 2M max
[...]
0   addr 4.000 GiB    size 8.000 MiB   access 0.069 hz age 2 m 53.546 s     df-passed 0 B
1   addr 4.008 GiB    size 12.000 MiB  access 0 hz   age 2 m 53.546 s       df-passed 0 B
2   addr 4.020 GiB    size 816.000 KiB access 1.383 hz age 3 h 4 m 37.999 s df-passed 2.000 MiB
[...]
11  addr 25.053 GiB   size 3.023 GiB   access 0 hz   age 3 m 8.008 s       df-passed 20.000 MiB
12  addr 28.075 GiB   size 123.578 MiB  access 0.069 hz age 3 m 8.008 s      df-passed 0 B
13  addr 28.196 GiB   size 185.367 M    access 0.069 hz age 3 m 8.008 s      df-passed 2.000 MiB
[...]
24  addr 64.780 GiB   size
memory bw estimate: 3.079 G
total size: 62.000 GiB  df-
```

This is not accessed for last 3 minutes

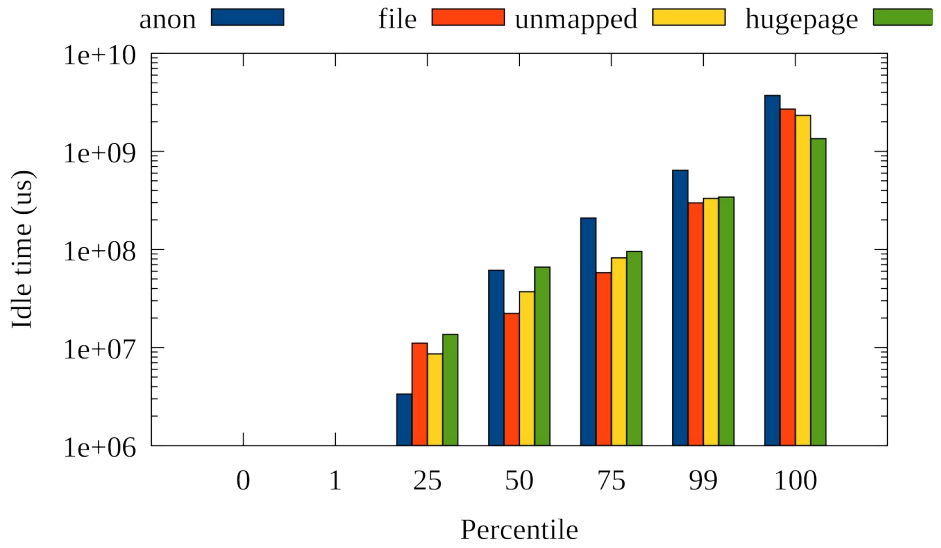
And it is using a 10 2MiB-size huge pages (we are doing ... BAD?)

Meta Contributions

- ‘hugepage_size’ filter: Usama Arif
- ‘active’ filter: Nhat Pham
- Virtual address space page level monitoring support: Yueyang Pan

Page Level Access in Real

- Files are hotter than anon
- Huge pages are hotter in general



Per-page type access pattern of a production workload

Supported Filter Types

- Backing content (file or anon)
- Belonging Cgroup
- PG_Young
- Folio size
- Whether on active LRU list

Future Works: Optimization

- Page level filter: expensive, unscalable
  - Not applicable in fleet level
- Workaround: Machine-level sampling
- Idea: page sampling, timeline TBD
  - Show your interest for prioritization
  - Or, send patches!

Fleet Wide Monitoring

Fleet Wide Monitoring TOC

- Monitoring intervals auto-tuning
- Format for Fleet-wide aggregation
- DAMON_STAT

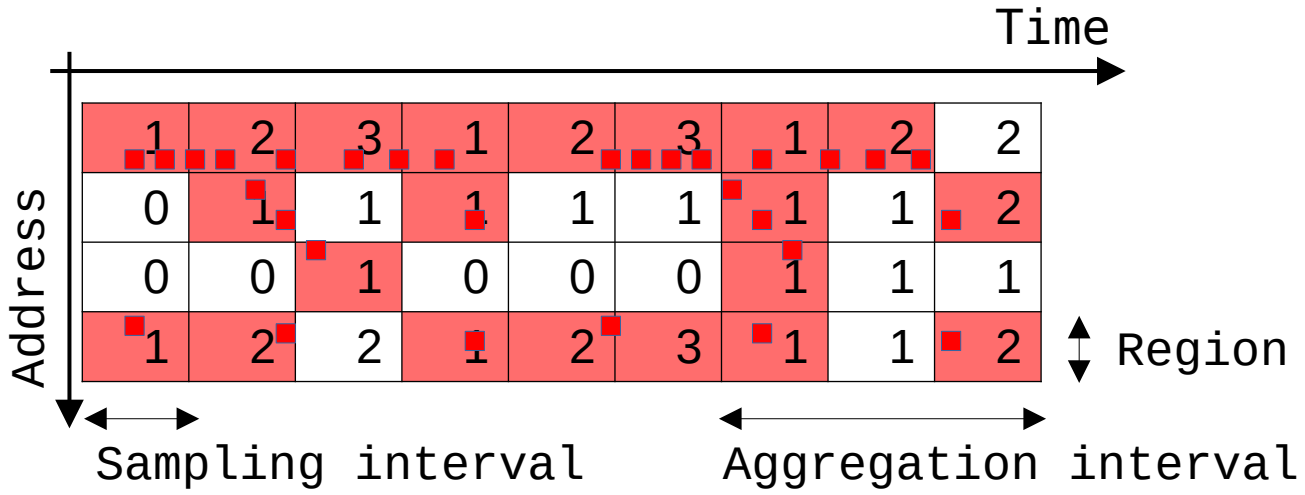
Monitoring Intervals Auto-tuning

Motivation: Why So Inaccurate?

- DAMON parameters need tuning
  - For each workload/system
  - Without tuning it looks only chaotic or boring
- Meta runs more than one workloads
  - No more tuning!

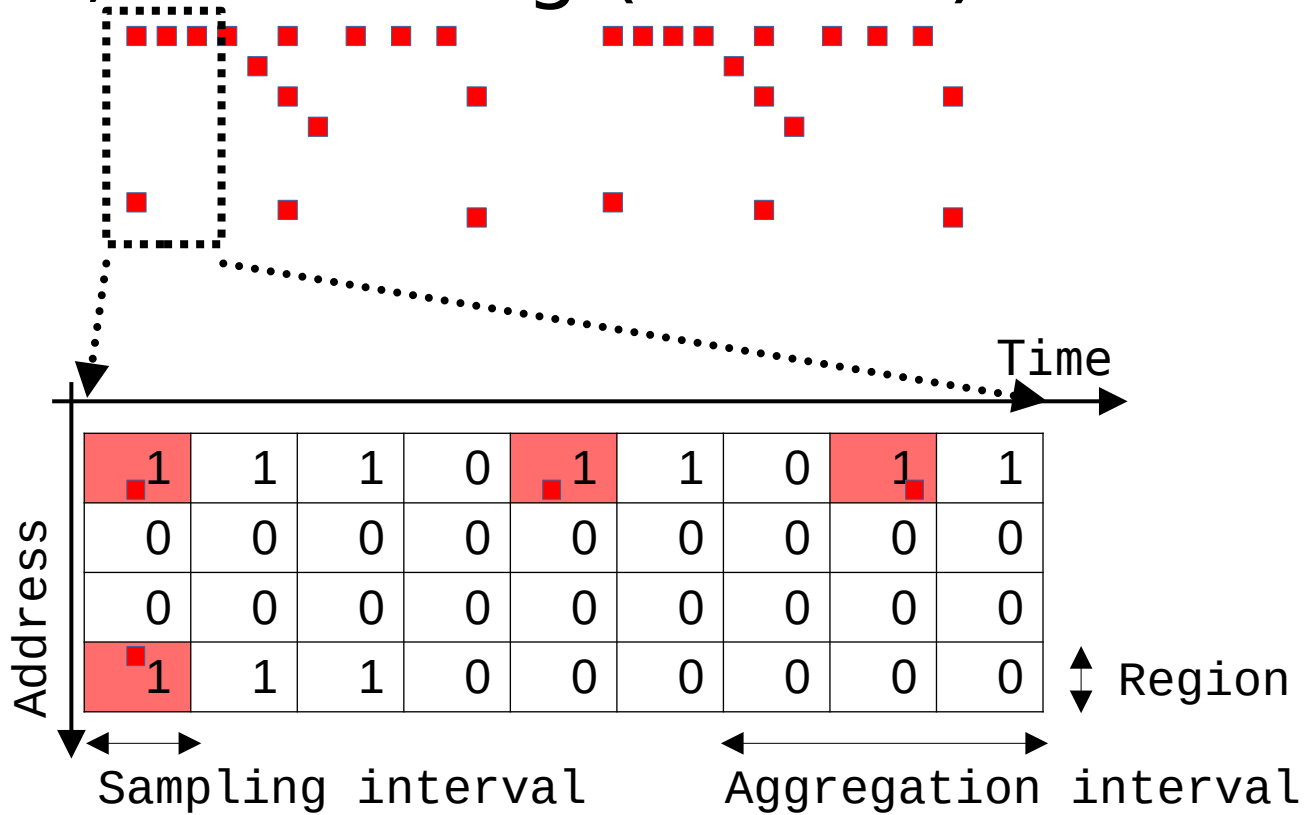
Supposed DAMON Snapshot

- Captures actionable amount of diversity



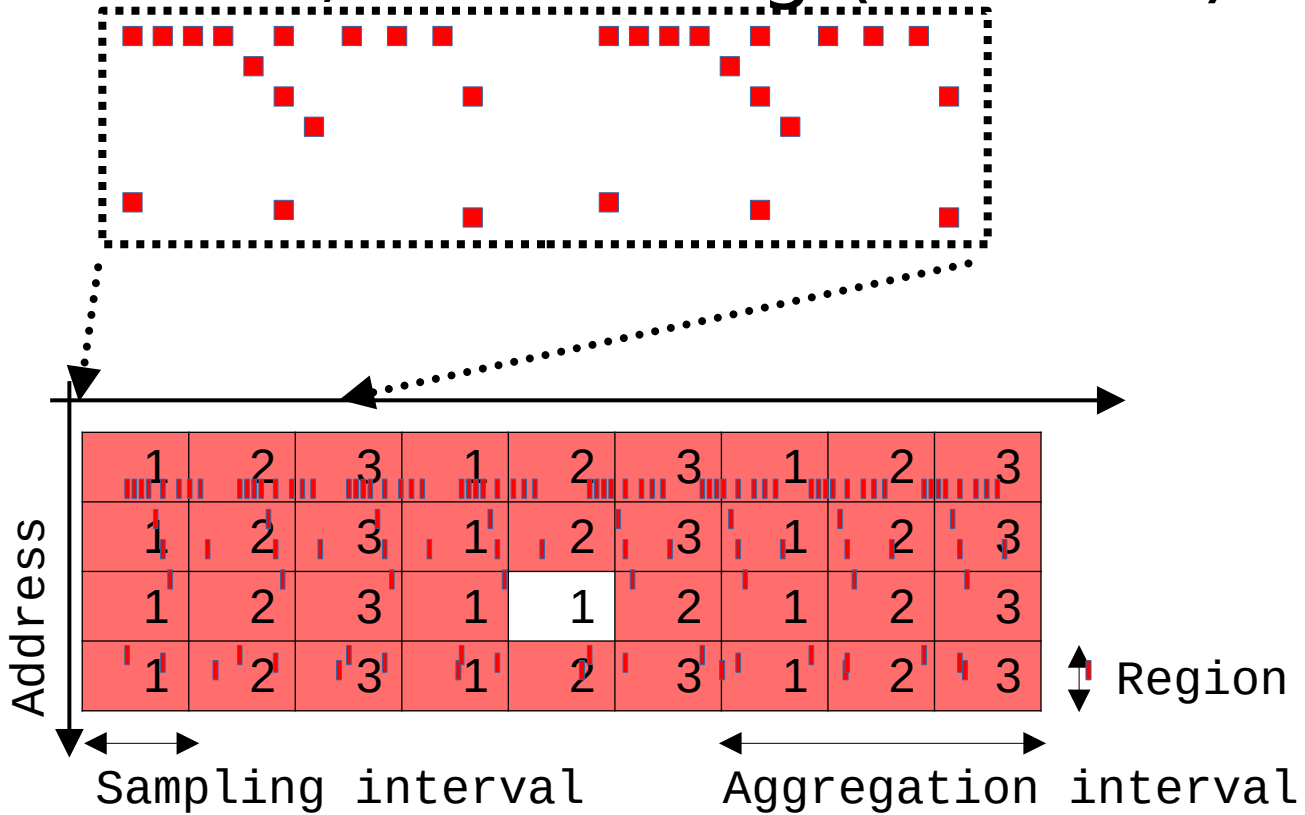
If Intervals Are Too Short

- Frozen, but boring (no Elsa)



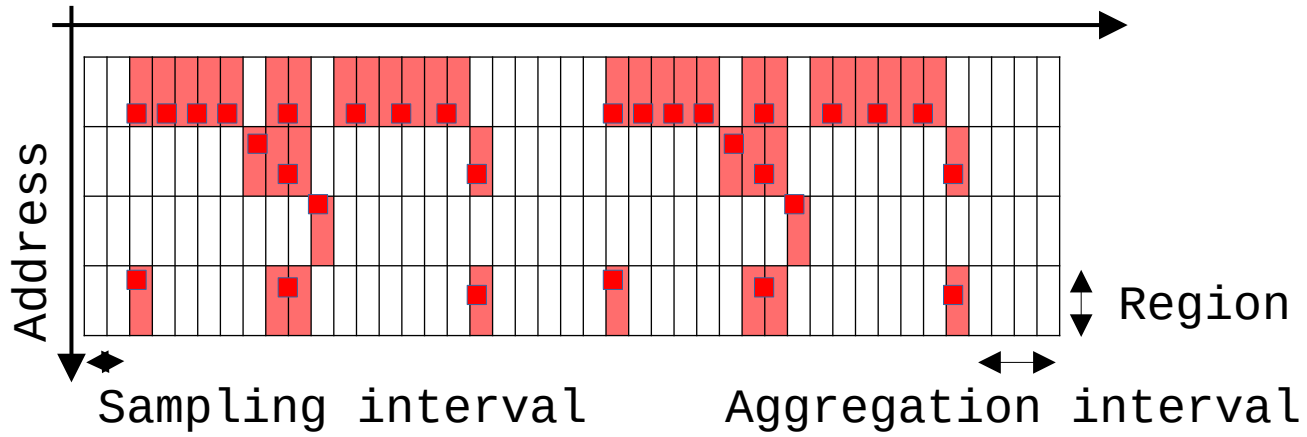
If Intervals Are Too Long

- Muspelheim, but boring (no Thor)



Too Short Sampling Interval

- Most sampling returns “negative”: unnecessary CPU cycle waste



Too Short Intervals in Real 1/2

- 5ms sampling, 100ms aggregation
- Frozen, without Elsa

```
# damo report access --sort_regions_by temperature
0   addr 16.052 GiB   size 5.985 GiB   access 0 %   age 5.900 s    # coldest
1   addr 22.037 GiB   size 6.029 GiB   access 0 %   age 5.300 s
2   addr 28.065 GiB   size 6.045 GiB   access 0 %   age 5.200 s
3   addr 10.069 GiB   size 5.983 GiB   access 0 %   age 4.500 s
4   addr 4.000 GiB    size 6.069 GiB   access 0 %   age 4.400 s
5   addr 62.008 GiB   size 3.992 GiB   access 0 %   age 3.700 s
6   addr 56.795 GiB   size 5.213 GiB   access 0 %   age 3.300 s
7   addr 39.393 GiB   size 6.096 GiB   access 0 %   age 2.800 s
8   addr 50.782 GiB   size 6.012 GiB   access 0 %   age 2.800 s
9   addr 34.111 GiB   size 5.282 GiB   access 0 %   age 2.300 s
10  addr 45.489 GiB   size 5.293 GiB   access 0 %   age 1.800 s    # hottest
total size: 62.000 GiB
```

Too Short Intervals in Real 2/2

- 5ms sampling, 100ms aggregation
- Frozen, without Elsa

```
# damo report access --style temperature-sz-hist
<temperature> <total size>
[-,590,000,000, -,549,000,000) 5.985 GiB  |**********            |
[-,549,000,000, -,508,000,000) 12.074 GiB |********************|
[-,508,000,000, -,467,000,000) 0 B         |                      |
[-,467,000,000, -,426,000,000) 12.052 GiB |********************|
[-,426,000,000, -,385,000,000) 0 B         |                      |
[-,385,000,000, -,344,000,000) 3.992 GiB  |*******               |
[-,344,000,000, -,303,000,000) 5.213 GiB  |*********             |
[-,303,000,000, -,262,000,000) 12.109 GiB |********************|
[-,262,000,000, -,221,000,000) 5.282 GiB  |*********             |
[-,221,000,000, -,180,000,000) 0 B         |                      |
[-,180,000,000, -,139,000,000) 5.293 GiB  |*********             |
```

Tuned Intervals in Real 1/2

- 400ms sampling, 8s aggregation
- “Look” actionable

```
# damo report access --sort_regions_by temperature
0   addr 64.492 GiB   size 1.508 GiB   access 0 %   age 6 m 48 s    # coldest
1   addr 21.749 GiB   size 5.674 GiB   access 0 %   age 6 m 8 s
2   addr 27.422 GiB   size 5.801 GiB   access 0 %   age 6 m
[...]
25  addr 6.615 GiB    size 297.531 MiB access 15 %  age 0 ns
26  addr 9.513 GiB    size 12.000 KiB  access 20 %  age 0 ns
27  addr 9.511 GiB    size 108.000 KiB access 25 %  age 0 ns
[...]
43  addr 58.321 GiB   size 4.000 KiB   access 100 % age 6 m 24 s
44  addr 9.512 GiB    size 4.000 KiB   access 100 % age 6 m 48 s
45  addr 58.106 GiB   size 4.000 KiB   access 100 % age 6 m 48 s    # hottest
total size: 62.000 GiB
```

Tuned Intervals in Real 2/2

- 400ms sampling, 8s aggregation
- “Look” actionable

```
# damo report access --style temperature-sz-hist
<temperature> <total size>
[-42,800,000,000, -33,479,999,000) 22.018 GiB |******************    |
[-33,479,999,000, -24,159,998,000) 27.090 GiB |**********************|
[-24,159,998,000, -14,839,997,000) 6.836 GiB  |******                |
[-14,839,997,000, -5,519,996,000)  6.056 GiB  |*****                 |
[-5,519,996,000, 3,800,005,000)    4.000 KiB  |*                     |
[3,800,005,000, 13,120,006,000)    0 B        |                      |
[13,120,006,000, 22,440,007,000)   0 B        |                      |
[22,440,007,000, 31,760,008,000)   0 B        |                      |
[31,760,008,000, 41,080,009,000)   0 B        |                      |
[41,080,009,000, 50,400,010,000)   0 B        |                      |
[50,400,010,000, 59,720,011,000)   4.000 KiB  |*                     |
```

Intervals Auto-tuning

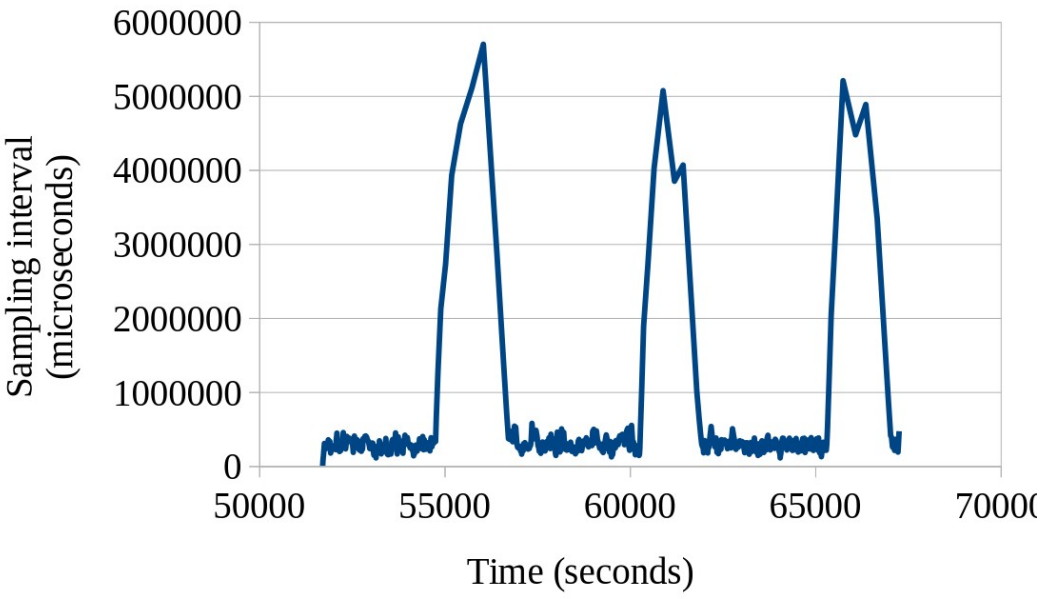
- Change Question
  - How to do? (mechanism) → What to achieve? (final goal, policy)
- Users specify
  - Desired amount of access events in each snapshot
  - Minimum and maximum sampling intervals
- Tune intervals for the desire using a feedback loop
  - Increase intervals if events in current snapshot < goal
  - Decrease intervals if events in current snapshot > goal

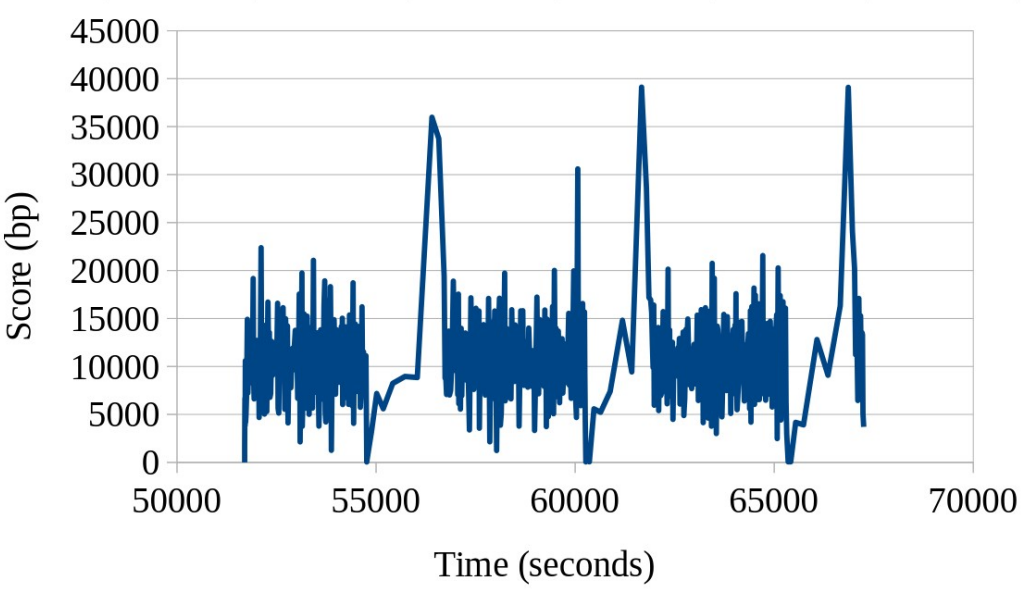
Parameters for Auto-tune

- Parameters for parameters auto-tuning
  - Ironic, but inevitable
  - Arguably easier to tune
- Default suggestion is available
  - Proven on real-world production workloads
  - Should be good enough to at least start

Auto-tuning in Real World

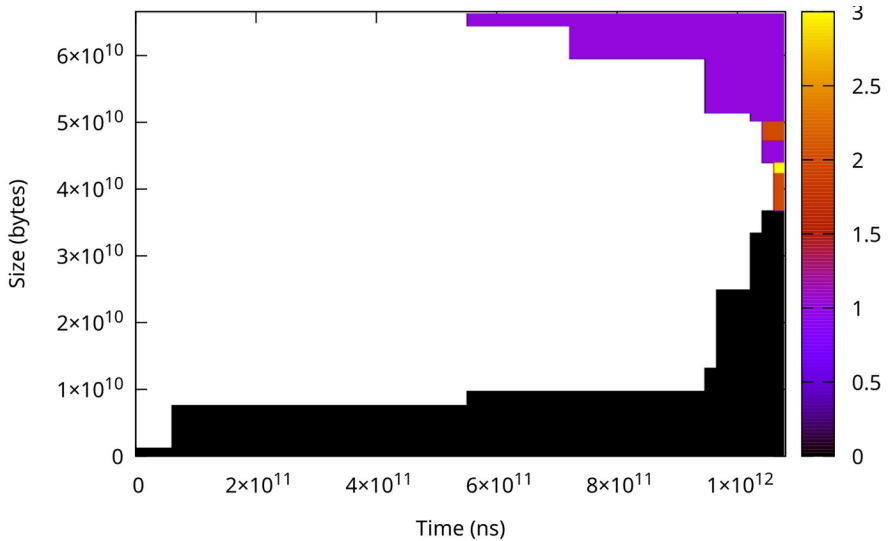
- Dynamically change and converge
  - As expected

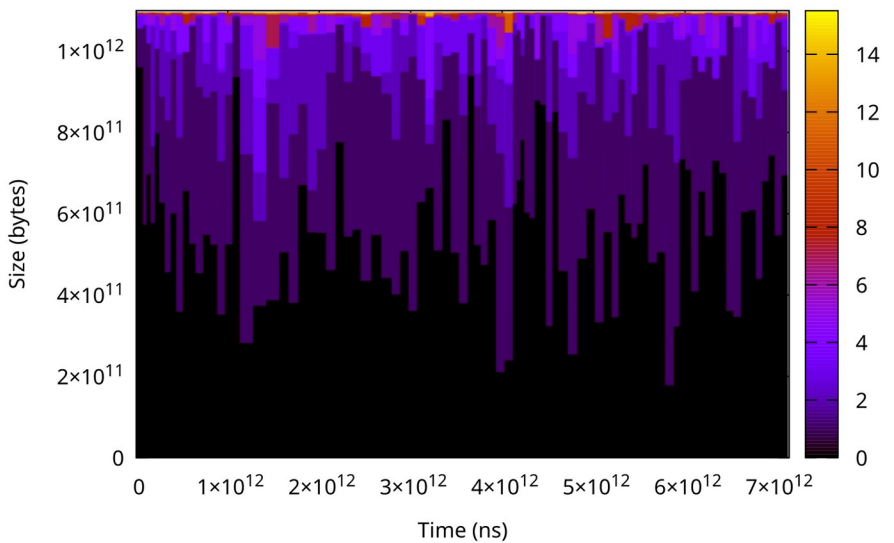




Auto-tuned Snapshots

- Meaningful access patterns on multiple workloads
- <0.1% single CPU time consumption





Format for Fleet-wide Aggregation

Motivation: Finer != Always Fine

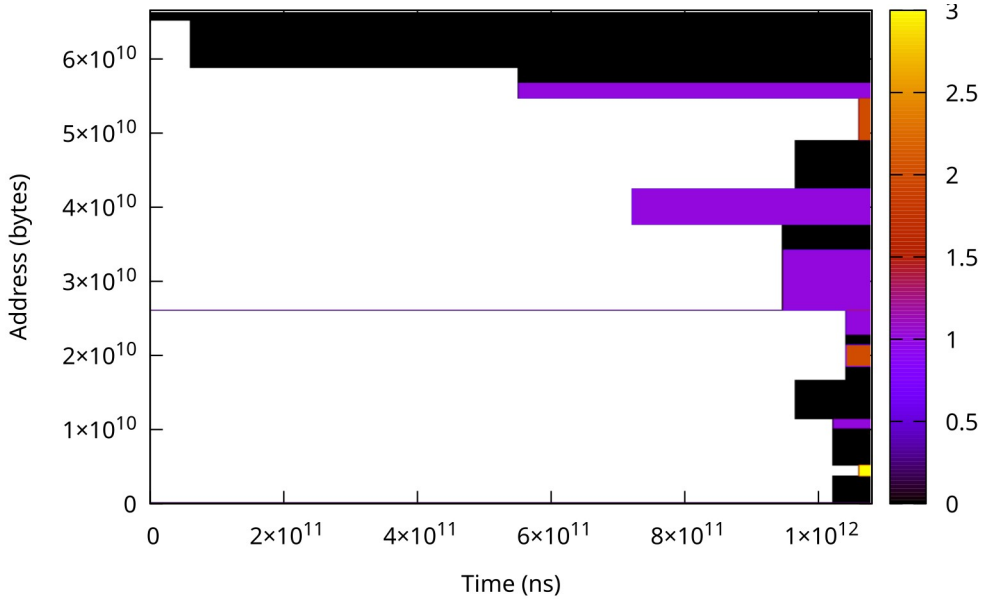
- Data from machines need aggregation
- DAMON provides
  location, frequency, stability, recency
  - Location only disturbs aggregation

```
$ sudo damo report access
[...]
0   addr 4.000 GiB    size 20.000 MiB  access 0 hz     age 2 m 53.546 s
1   addr 4.020 GiB    size 2.000 MiB   access 1.383 hz age 3 h 4 m 37.999 s
2   addr 4.021 GiB    size 6.046 GiB   access 0.069 hz age 1 m 26.773 s
[...]
```

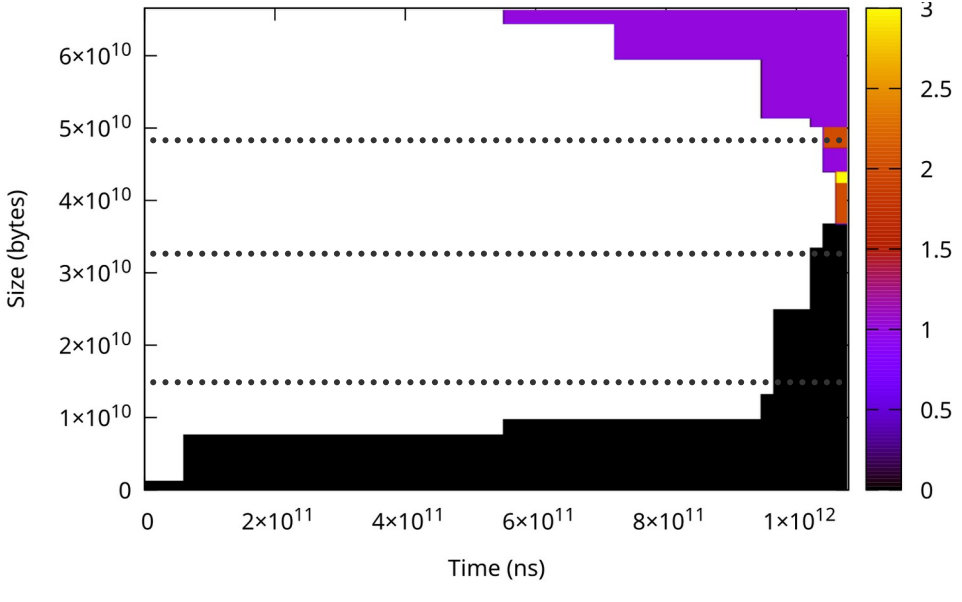
Idle Time Percentiles: Concept

- Idle time: how long it was not accessed
- Percentile: that of the statistics

Unsorted snapshot



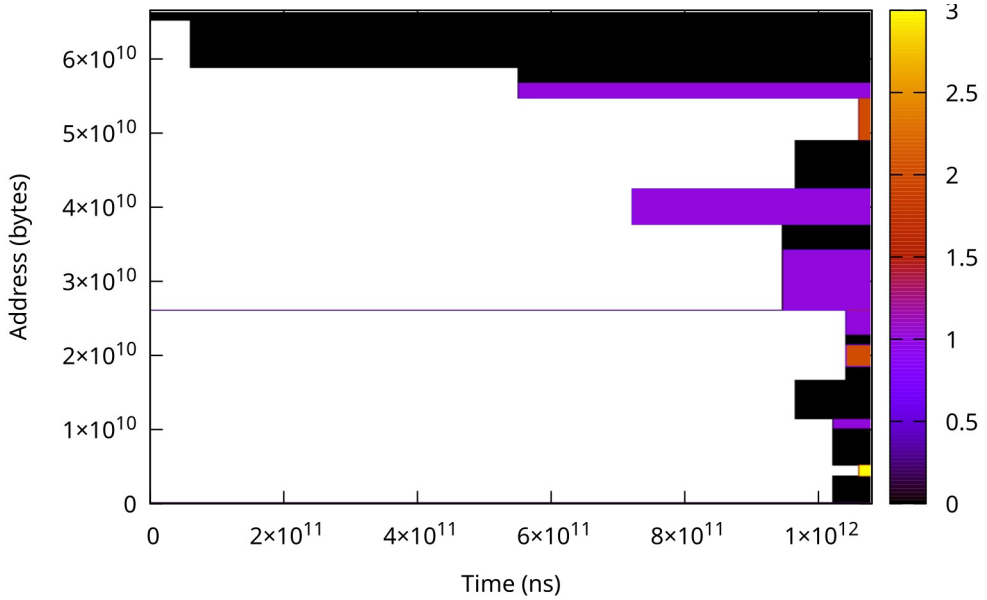
Sorted by access frequency



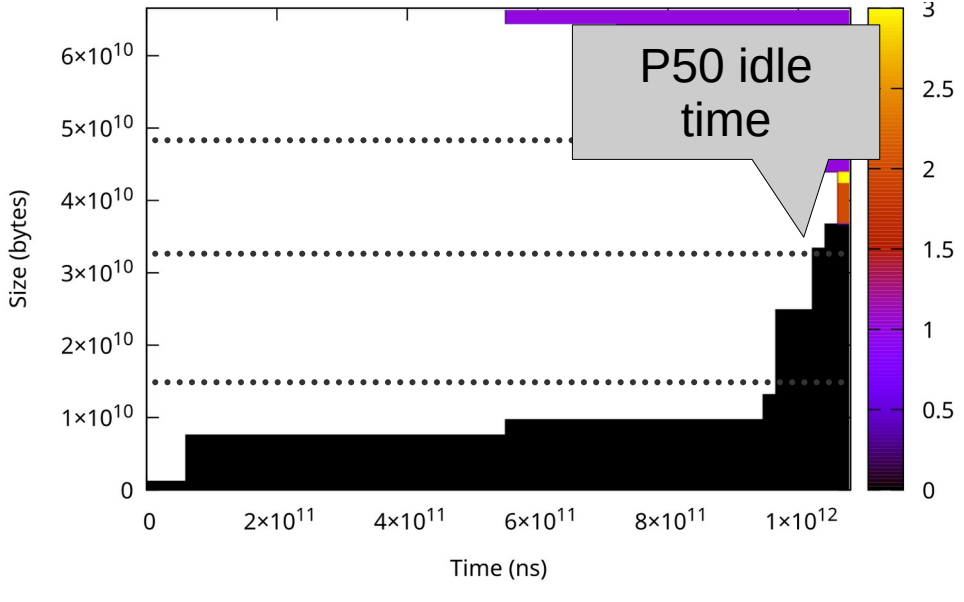
Idle Time Percentiles: Concept

- Idle time: how long it was not accessed
- Percentile: that of the statistics

Unsorted snapshot



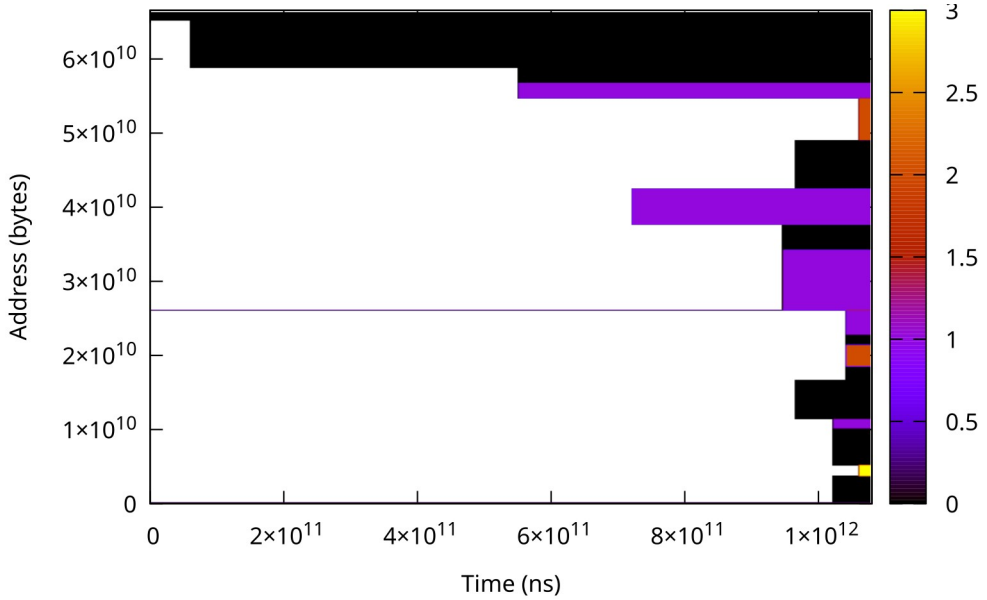
Sorted by access frequency



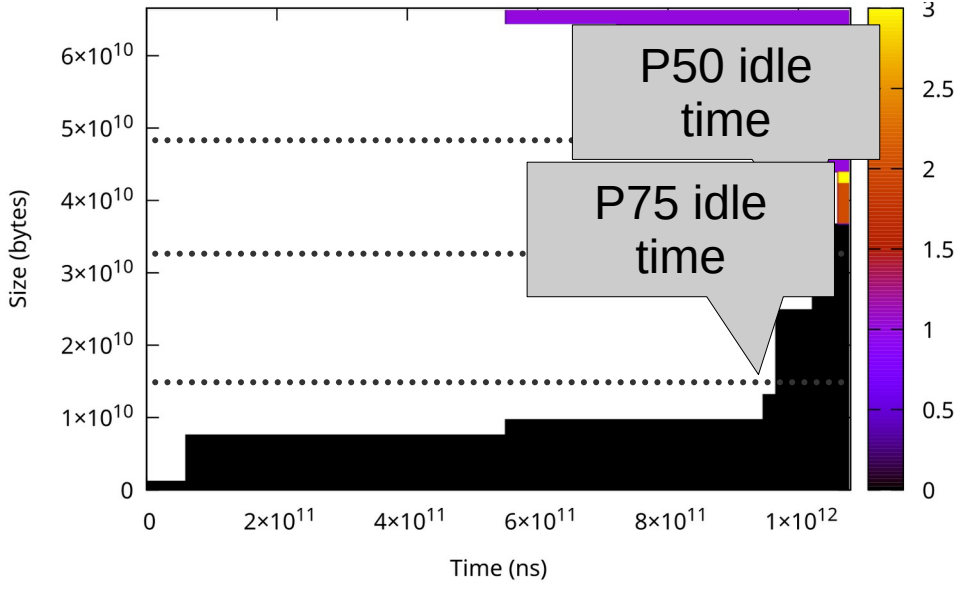
Idle Time Percentiles: Concept

- Idle time: how long it was not accessed
- Percentile: that of the statistics

Unsorted snapshot



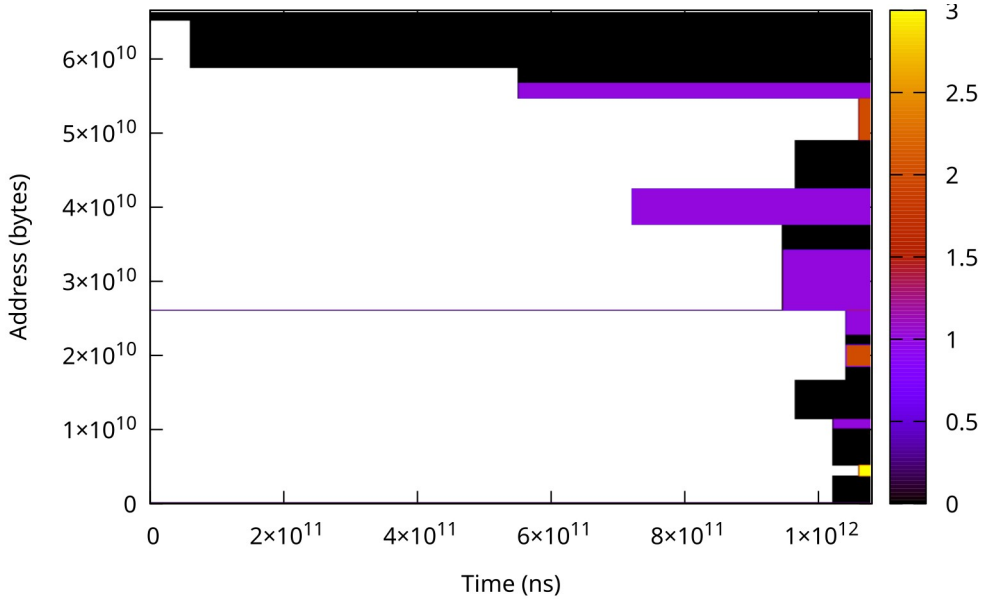
Sorted by access frequency



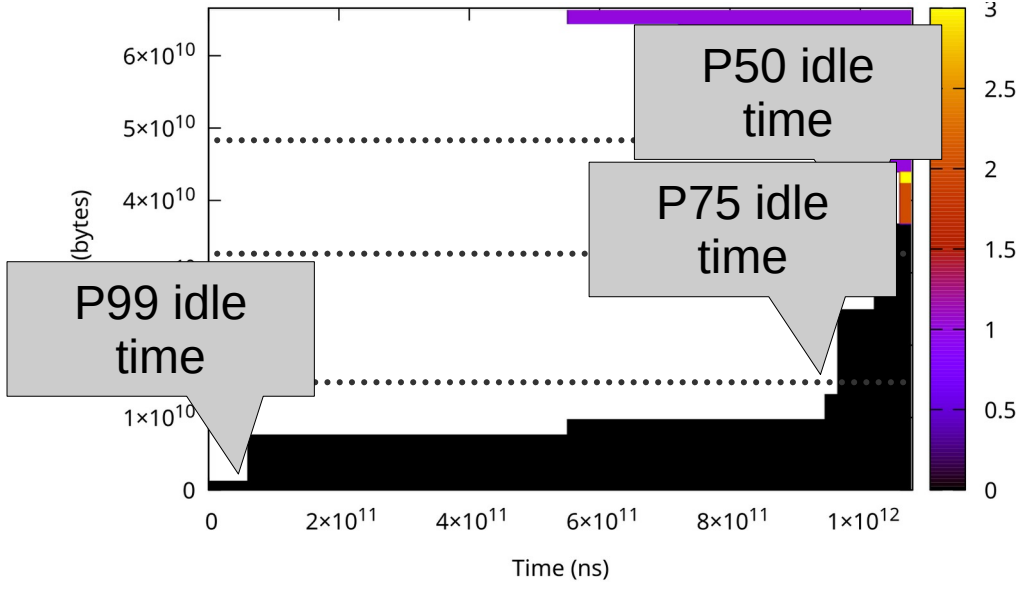
Idle Time Percentiles: Concept

- Idle time: how long it was not accessed
- Percentile: that of the statistics

Unsorted snapshot

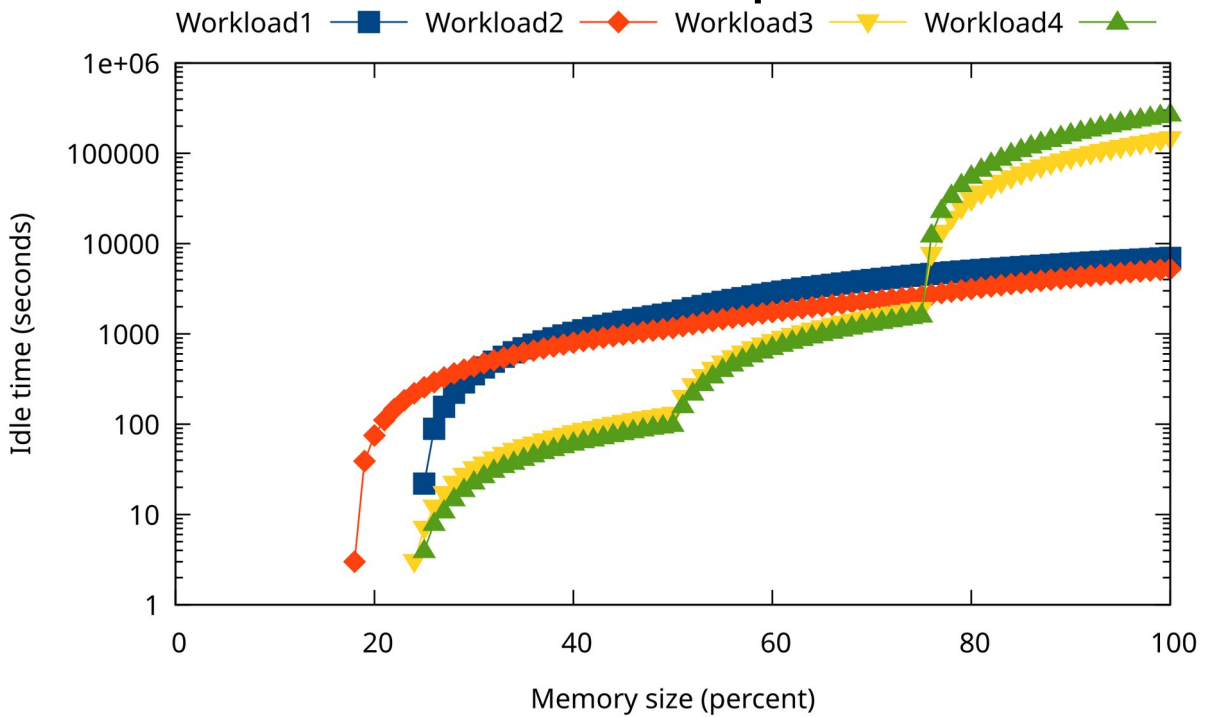


Sorted by access frequency



Idle Time Percentiles in Real

- Note: Y-axis in logscale
- Show common and different patterns of workloads

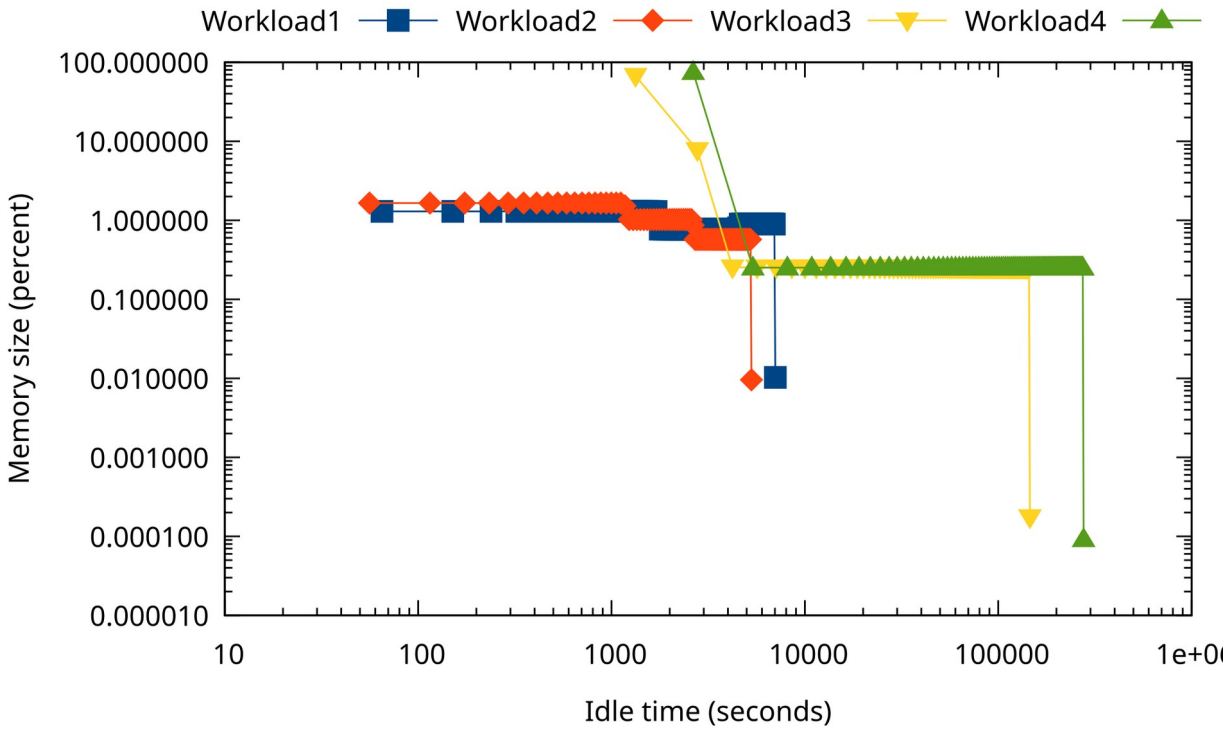


Cold Memory Tail

- X/Y-axis inversion of idle time percentiles
- X axis: Idle time
- Y axis: Size of memory of the idle time

Cold Memory Tail in Real

- Note: Both X/Y-axis in logscale



DAMON_STAT

Motivation: Usability

- DAMON is in Kernel
- Need a user-space agent
  - Run DAMON with the features
  - Collect the monitoring outputs
  - Send the outputs to a server

DAMON_STAT: A kernel module

- Run auto-tuned DAMON
- Expose idle time percentiles and estimated memory bandwidth with files
- Can enabled at runtime, boot time, build time
- Upstream-ed v6.17

Interface: Module Params

- DAMON_STAT_ENABLED_DEFAULT for build time enable

```
$ cd /sys/module/damon_stat/parameters/
$ ls
aggr_interval_us  enabled  estimated_memory_bandwidth
memory_idle_ms_percentiles
$ sudo cat estimated_memory_bandwidth
3860000
$ sudo cat memory_idle_ms_percentiles
-25600,25600,25600,25600,25600,[...]128000,128000
```

How To Do Fleet Wide Monitor

- Set DAMON_STAT_ENABLED_DEFAULT=y
- Modify your monitoring agent
  - Read idle_time_percdentiles file
  - Send the content to the server

Wrapup

- Meta developed and upstreamed DAMON features for
  - page level and fleet wide access monitoring
- Feel free to use and
  please feedback DAMON

Questions?

- Community is there for you, too
  - Mailing list: damon@lists.linux.dev
  - Project website: https://damonitor.github.io/
  - Maintainer email: sj@kernel.org
  - DAMON Beer/Coffee/Tea Meetup