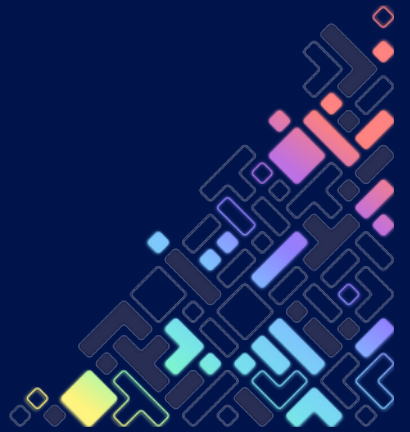
LINUXCON@ OPEN SOURCE SUMMIT NORTH AMERICA

THE LINUX FOUNDATION

DAMO[N,S]?: Implementing Self-Driven Data Access-Aware Efficient Linux System

SeongJae Park <sj@kernel.org>

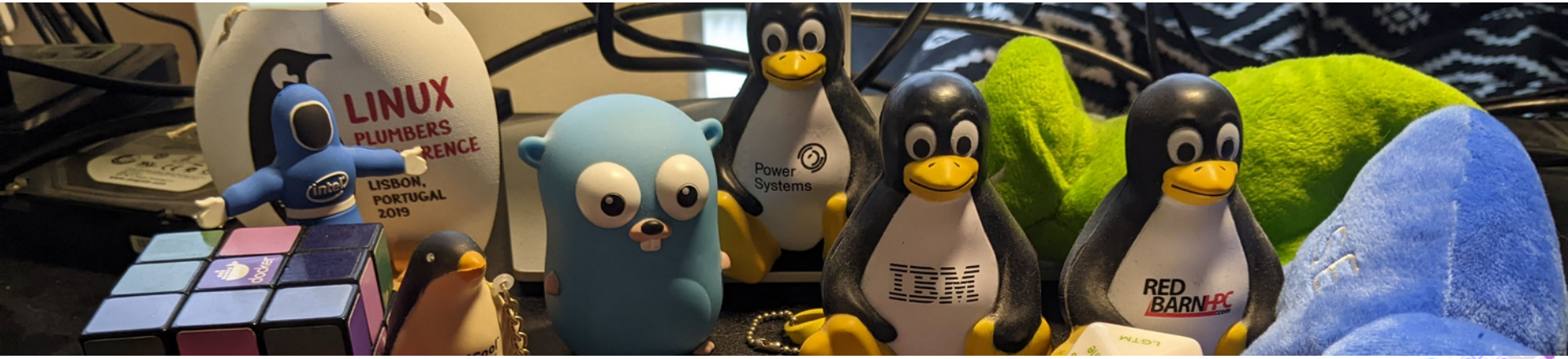


THE LINUX FOUNDATION

#ossummit #linux #kernel #damon

From: SeongJae Park <sj@kernel.org>

- Call me SJ
- Working on AWS
- Maintaining Linux kernel DAMON subsystem and its user-space tool, DAMO



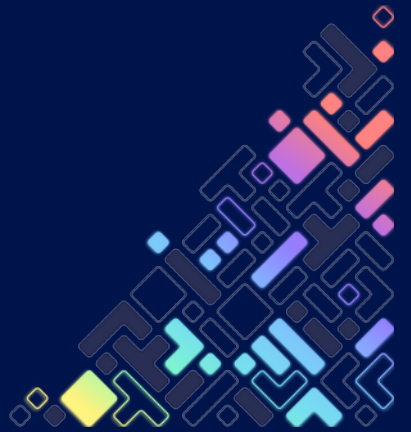
Notices

- The views expressed herein are those of the speaker;
- they do not reflect the views of his employers
- Examples on these slides are made with
  DAMO of commit 3a9b8e821bef (v2.3.1+) and Linux kernel v6.9-rc4

Overview

- Importance and Difficulties
- Generating Accesses Heatmap
- Online Access-aware System Operations
- DAMON Community
- Summary
- QnA

Importance and Difficulties



Increasing Demands and Complexity

- Memory demands are increasing
- New H/W will arrive as a new hierarchy, not a drop-in replacement
- Access-aware system operations can be helpful

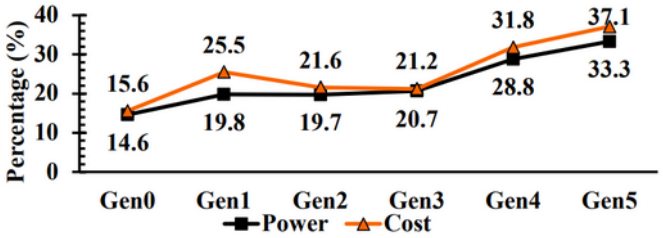


Figure 3: Memory as a percentage of rack TCO and power across different hardware generations of Meta.

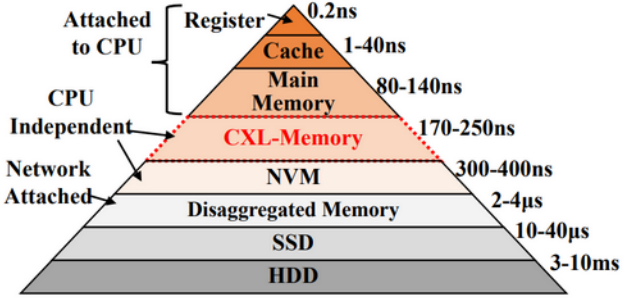
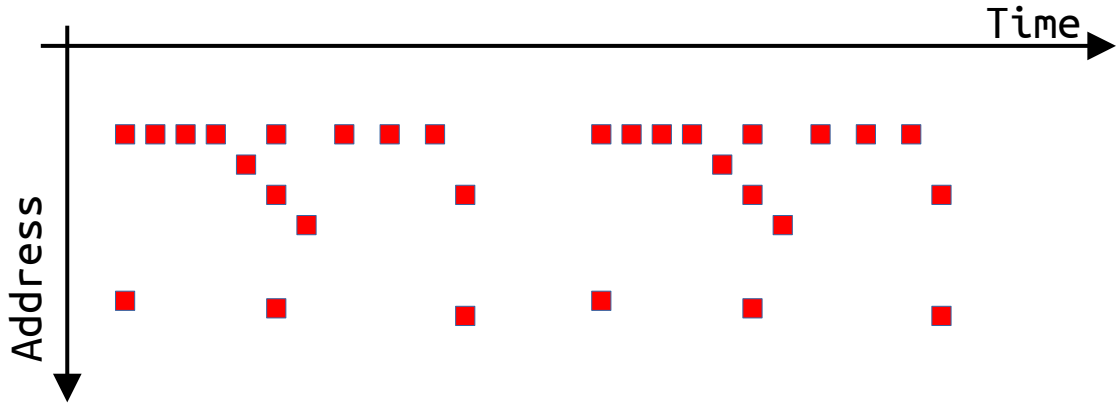


Figure 2: Latency characteristics of memory technologies.

Retrieved from “TPP: Transparent Page Placement for CXL-Enabled Tiered-Memory”, https://dl.acm.org/doi/pdf/10.1145/3582016.3582063

Data Accesses: Events on Space/Time of Memory

- Looks like a sort of a heatmap

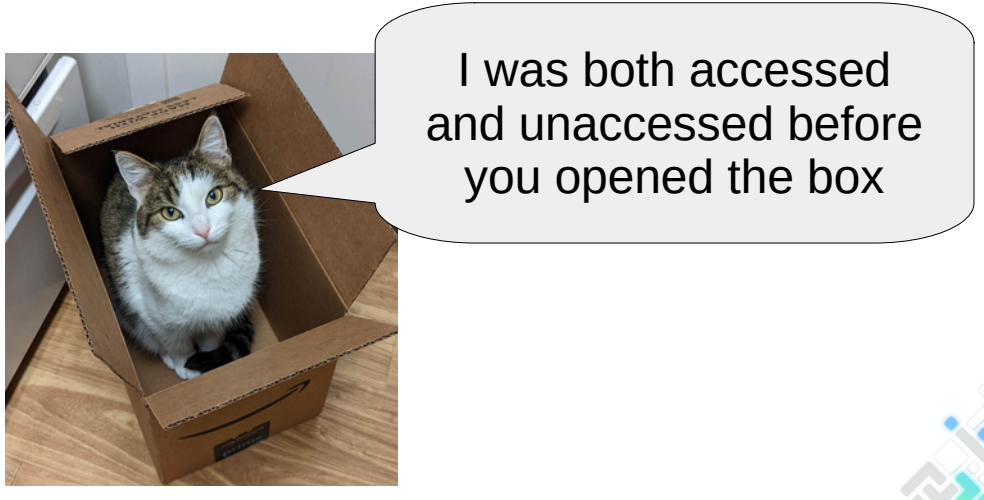


Ideal Data Access MONitor: Precise, Complete, Light

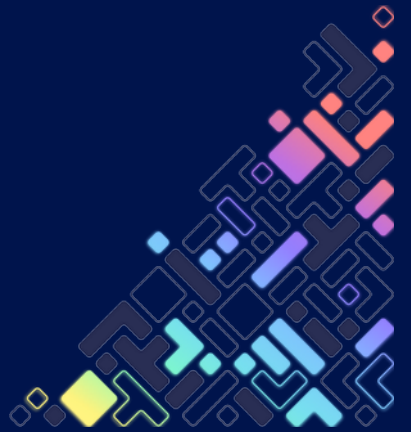
- Time granularity: CPU cycle / # CPUs (or, speed of light)
- Space granularity: bit (or, electron)
- Keep complete history (from Bigbang)
- Lightweight enough for general system operations

Physics: “It May Not That Easy”

- Observing/memorizing memory accesses via memory accesses
- Some of observer effects and the uncertainty principal may apply
- O(memory size)/O(memory size * total monitoring time) time/space overhead
  - Finite speed of light and atomic nature of matter



Generating Accesses Heatmap



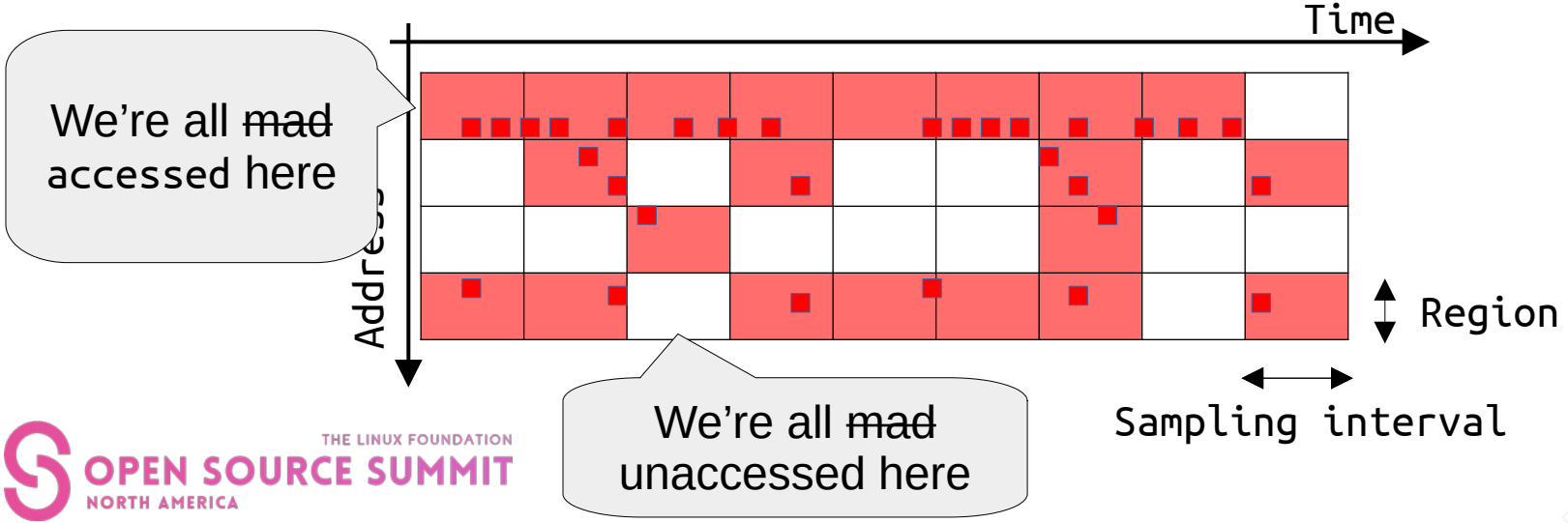
Region: Access Monitoring Unit

- Memory region: A sub-area of the memory's space-time
- Collection of elements that having similar access pattern
- Access check of one element per region is enough
- e.g., "This page is accessed within last 1 second; a cacheline is checked"

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

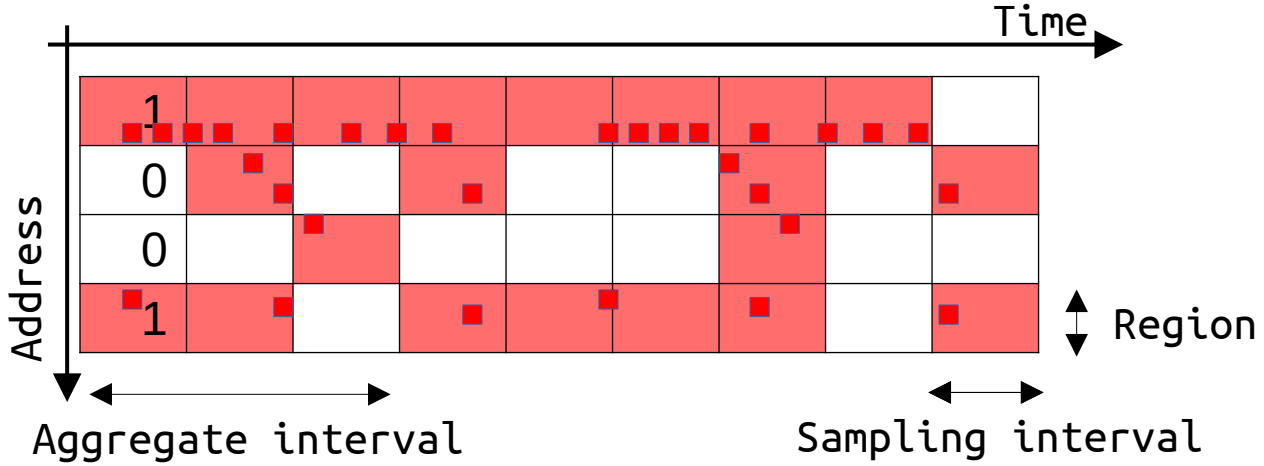
Fixed Space/Time Granularity, <=1 Access Frequency

- Sort of periodic fixed granularity idleness monitoring
- Time overhead: "*memory size* / space granularity"
- Space overhead: "*memory size* * *monitoring time* / time granularity"
- Reduced, but still ruled by memory size and total monitoring time



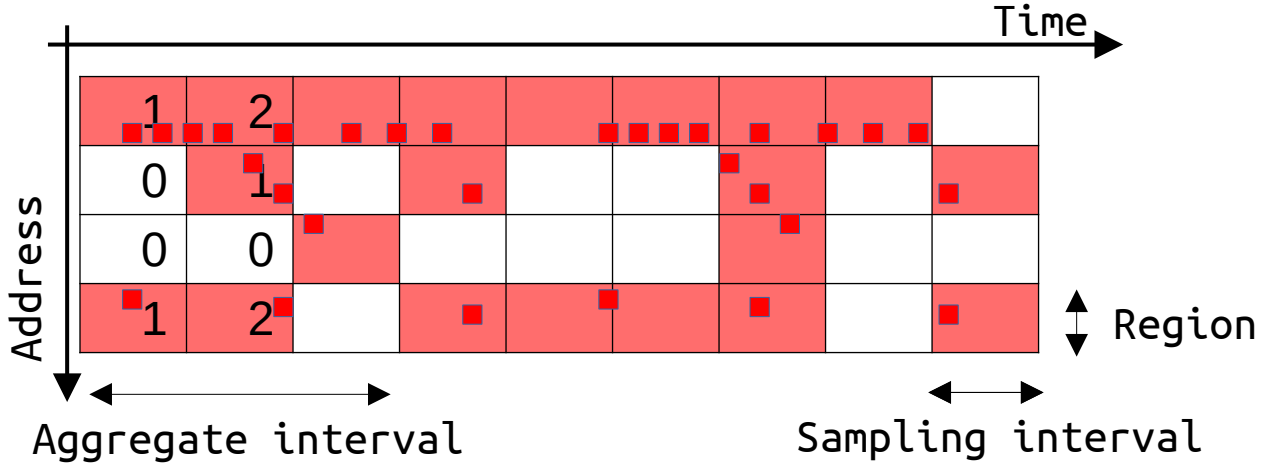
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate access checks via per-region counter



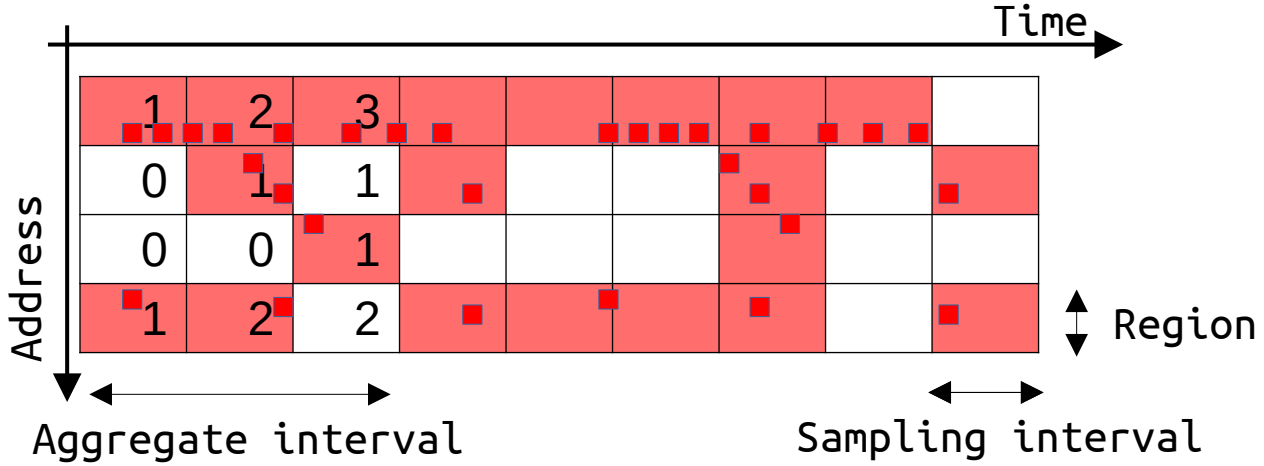
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate access checks via per-region counter



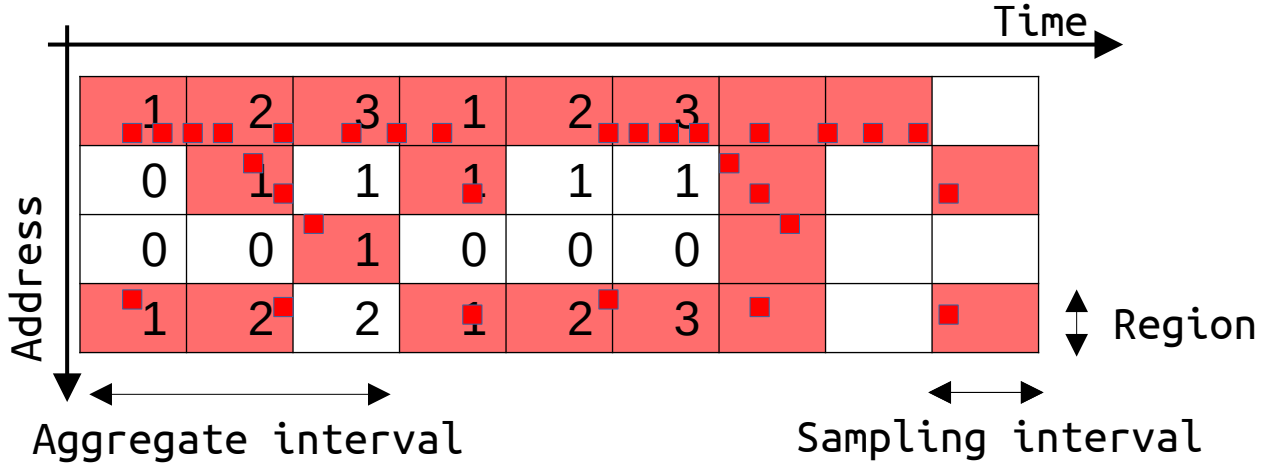
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate access checks via per-region counter



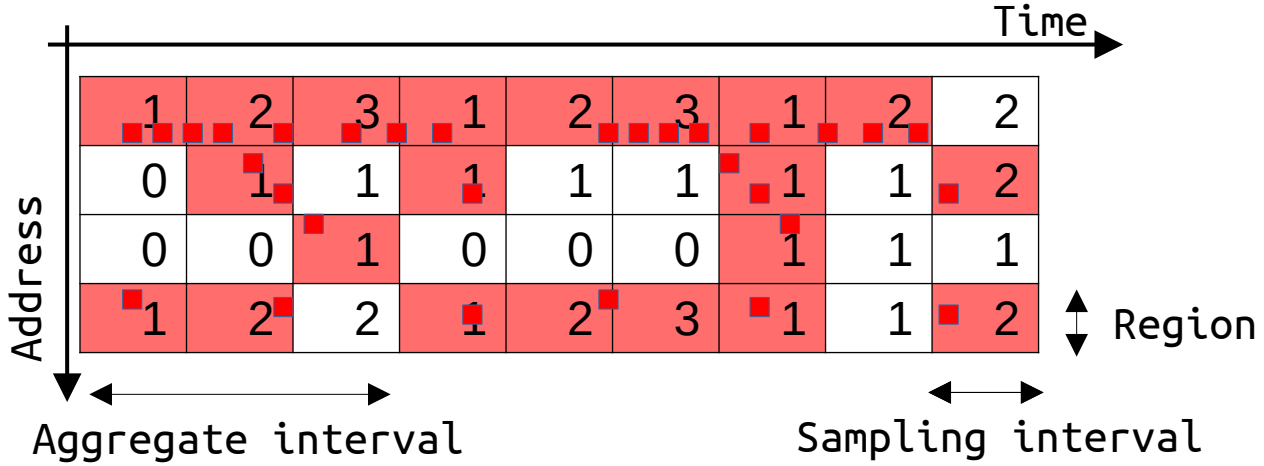
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate access checks via per-region counter



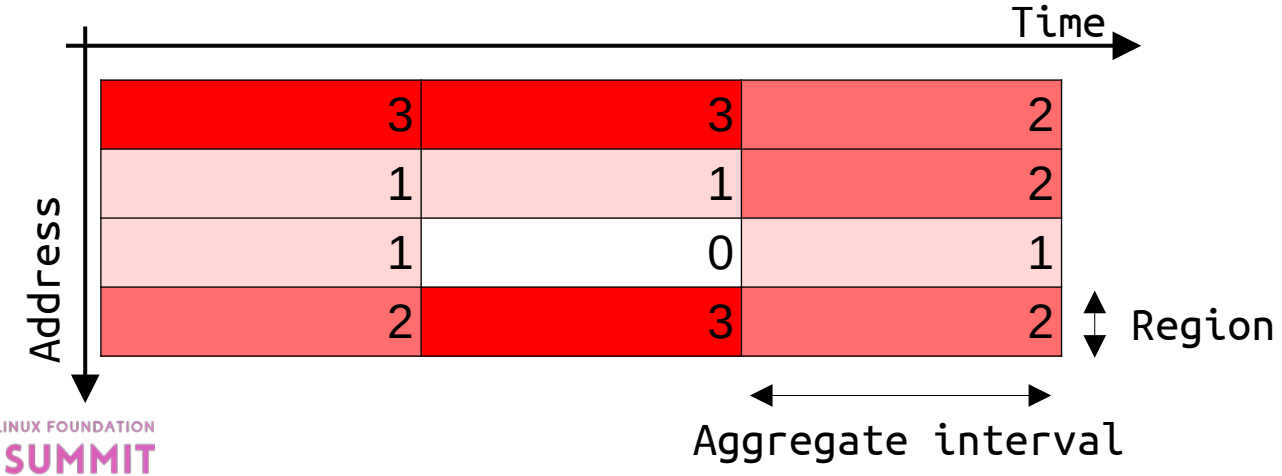
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate access checks via per-region counter



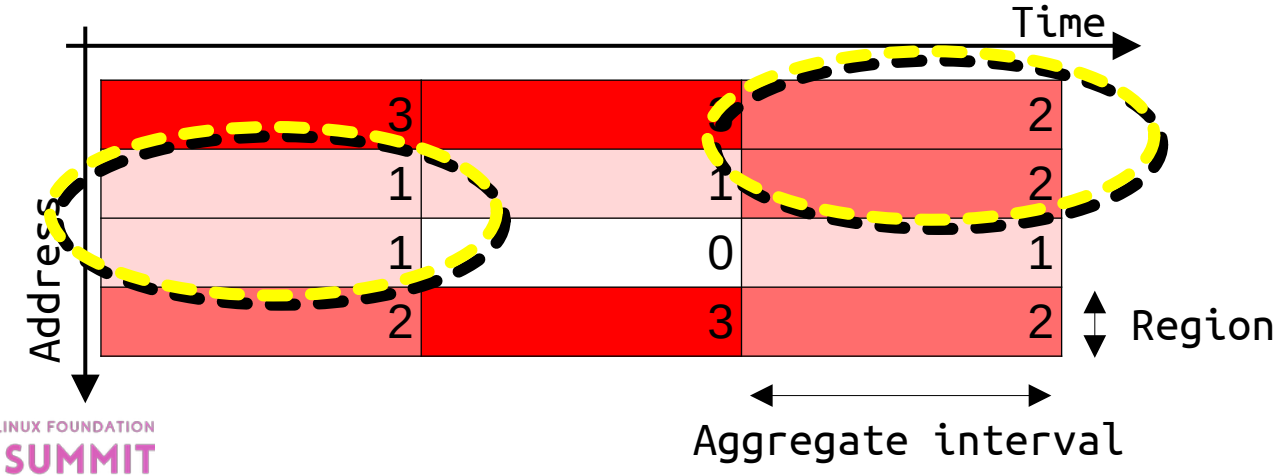
Fixed Space/Time Granularity, <=N Access Frequency

- Accumulate access checks via per-region counter
- Reduce space overhead to “1/N”
- Still, O(memory size * total monitoring time)



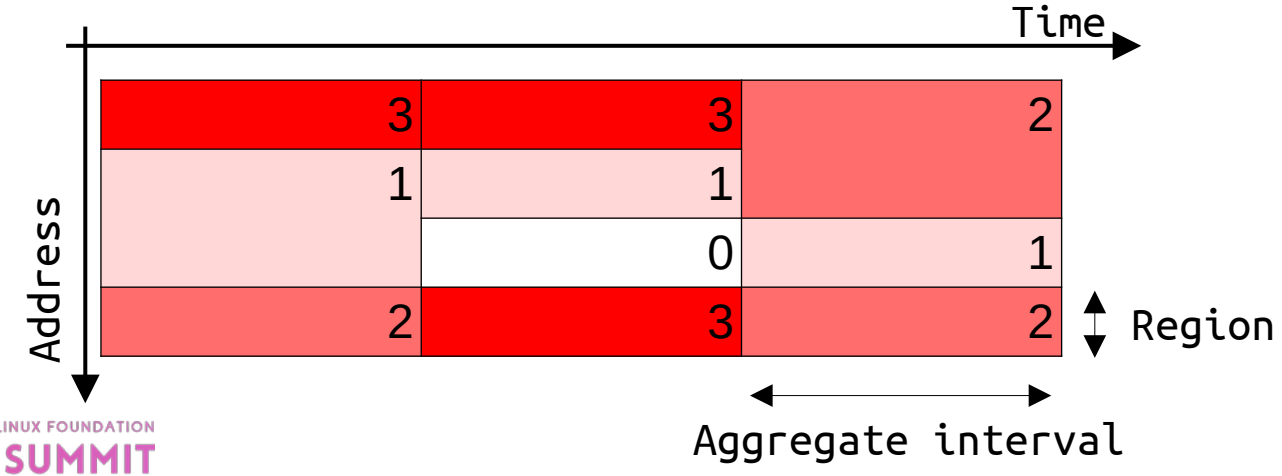
Inefficiency of Fixed Space Granularity

- Adjacent regions of similar hotness



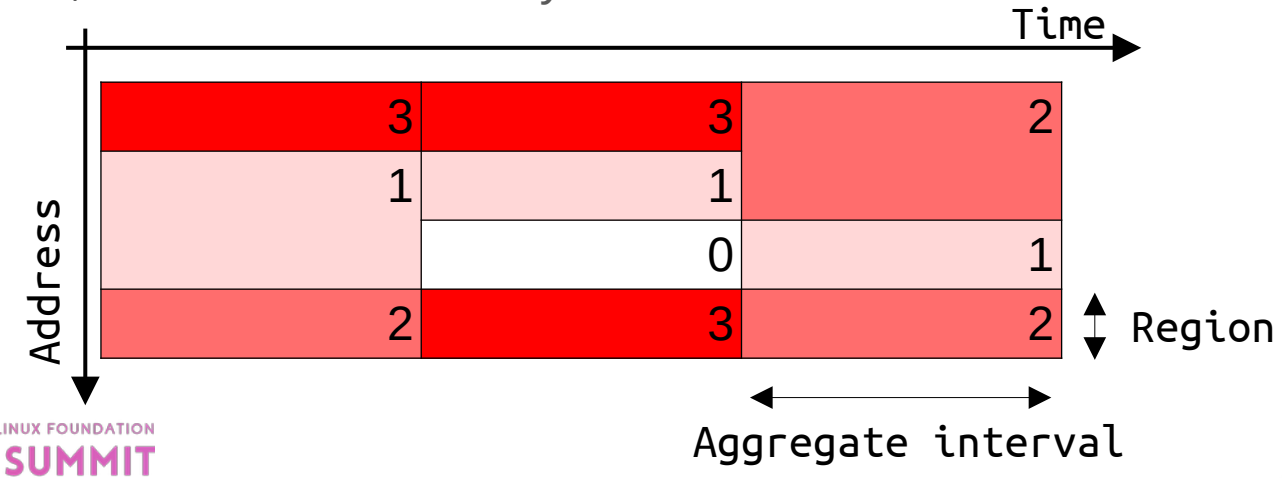
Dynamic Space Granularity: Mechanisms

- Merge the wasteful regions and randomly split region
  - The number of region == number of different access patterns
- Let user set *min/max number of regions*



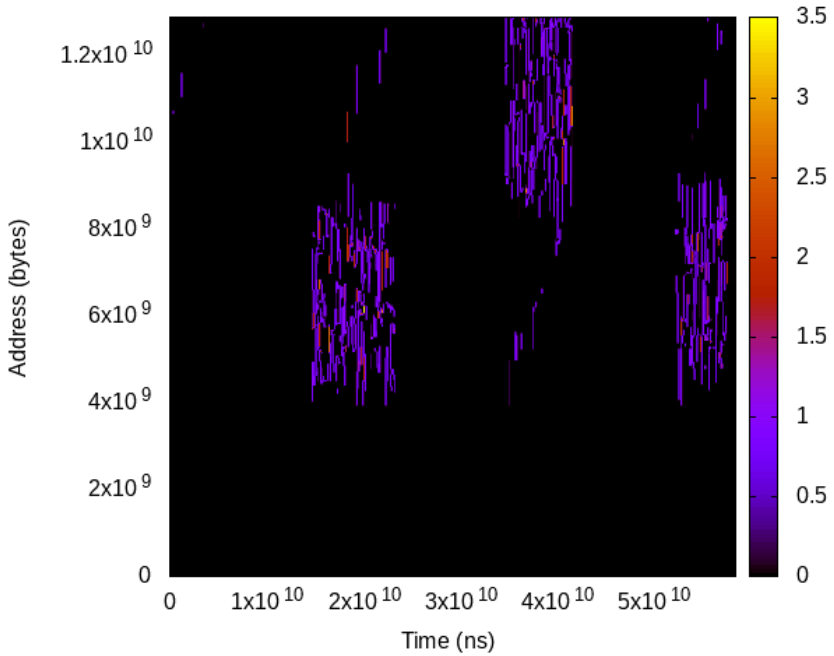
Dynamic Space Granularity: Overhead/Accuracy

- Time overhead: `min(different access patterns, max number of regions)`
  - No more ruled by arbitrary memory size
- Minimum accuracy: `min number of regions`
  - Best-effort, lower-bound accuracy



Output: Best-effort Controllable Accesses Heatmap

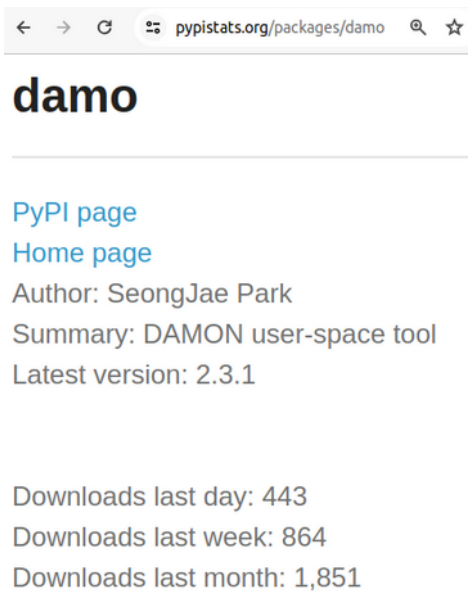
- How we can get it? DAMO



Heatmap of “Splash2x.fft”

DAMO: Data Access Monitoring Operator

- A Python3 program
- Provides the heatmap
- Packaged for PyPI and various distros
- Simple `git` cloning also works



Packaging status	
AUR	2.2.7
Debian 13	2.2.9
Debian Unstable	2.3.1
Devuan Unstable	2.3.1
EPEL 9	2.2.7
Fedora 37	1.9.8
Fedora 38	2.2.8
Fedora 39	2.2.8
Fedora Rawhide	2.3.1
Kali Linux Rolling	2.2.9
PyPI	2.3.1
Raspbian Testing	2.1.5
Ubuntu 23.10	1.8.7
Ubuntu 24.04	2.2.4

The images are captured on 2024-04-17

DAMON: Data Access MONitor (1/2)

- A Linux kernel subsystem
- Implement the heatmap generating mechanism;
  DAMO is a DAMON user-space tool
- Merged into the mainline from v5.15
- Backported and enabled on multiple distros

DAMON: I

- A Linux ker
- Implement
  DAMO is a
- Merged into
- Backported

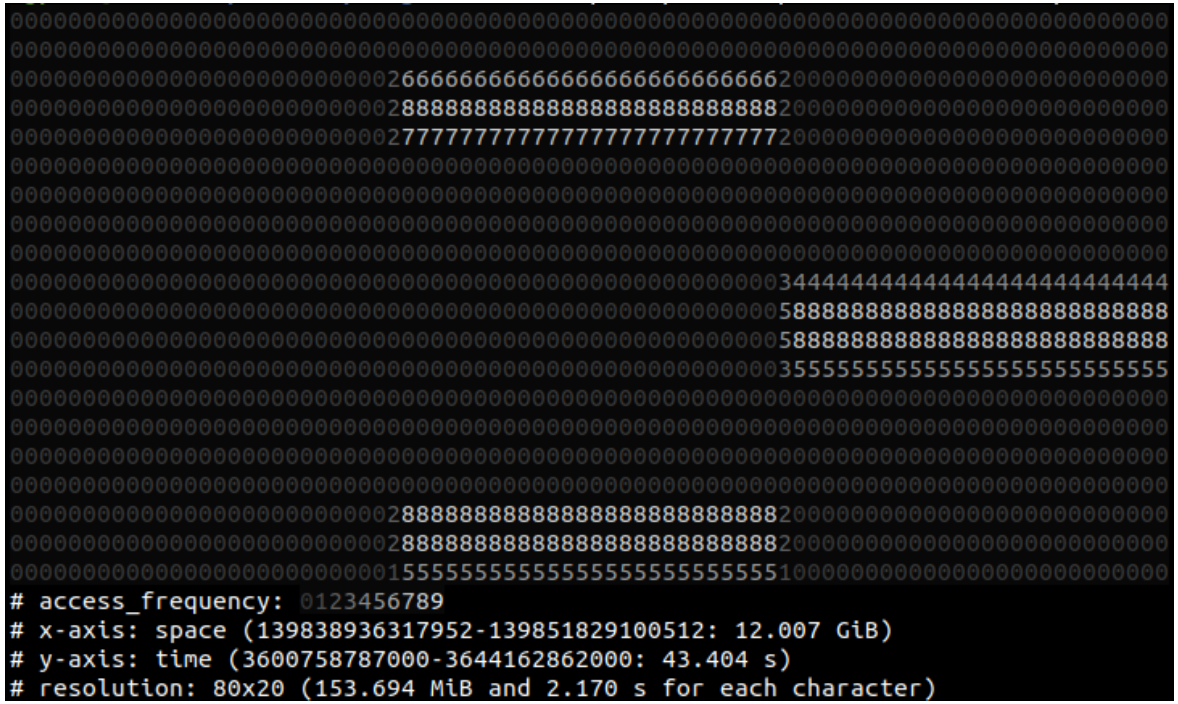
Distribution	DAMON [x]	UTS_RELEASE [x]
Android 12 (5.10) aarch64	y	5.10.205
Android 13 (5.10) aarch64	y	5.10.205
Android 13 (5.15) aarch64	y	5.15.144
Android 14 (5.15) aarch64	y	5.15.144
Android 14 (6.1) aarch64	y	6.1.68
Arch x86_64	y	6.8.5-arch1
CentOS 7 x86_64	—	3.10.0-1160.114.2.el7.x86_64
CentOS 8 Stream aarch64	y	4.18.0-552.el8.aarch64
CentOS 8 Stream x86_64	y	4.18.0-552.el8.x86_64
CentOS 9 Stream aarch64	y	5.14.0-435.el9.aarch64
Fedora Asahi Remix 40 aarch64	y	6.6.3-414.asahi.fc40.aarch64
Fedora Asahi Remix Rawhide aarch64	y	6.6.3-414.asahi.fc41.aarch64
Oracle Linux 7 (Red Hat Compatible) x86_64	—	3.10.0-1160.114.2.0.1.el7.x86_64
Oracle Linux 7 (UEK 4) x86_64	—	4.1.12-124.84.2.el7uek.x86_64
Oracle Linux 7 (UEK 5) aarch64	—	4.14.35-2047.535.2.1.el7uek.aarch64
Oracle Linux 7 (UEK 5) x86_64	—	4.14.35-2047.535.2.1.el7uek.x86_64
Oracle Linux 7 (UEK 6) aarch64	—	5.4.17-2136.330.7.1.el7uek.aarch64
Oracle Linux 7 (UEK 6) x86_64	—	5.4.17-2136.330.7.1.el7uek.x86_64
Oracle Linux 8 (Red Hat Compatible) x86_64	y	4.18.0-513.24.1.el8_9.x86_64
Oracle Linux 8 (UEK 6) aarch64	—	5.4.17-2136.330.7.1.el8uek.aarch64
Oracle Linux 8 (UEK 6) x86_64	—	5.4.17-2136.330.7.1.el8uek.x86_64
Oracle Linux 8 (UEK 7) aarch64	y	5.15.0-205.149.5.1.el8uek.aarch64

The images are captured on 2024-04-16

DAMO Demo: Generating The Heatmap (1/2)

```
# damo record "$HOME/parsec3_on_ubuntu/run.sh splash2x.fft" \
        --include_child_tasks --profile -footprint
[...]
real    50.489s
[...]
# damo report heats --heatmap stdout
```

DAMO Demo: Generating The Heatmap (2/2)



DAMO Demo: Profiling Memory Usage/Accesses (1/2)

```
# damo report footprints rss
[...]
# avr:  10.005 GiB
[...]
```


DAMO Demo: Profiling Memory Usage/Accesses (2/2)

```
# damo report footprints rss
[...]
# avr:  10.005 GiB
[...]
# damo report wss --collapse_targets
[...]
# avr:  1.872 GiB
[...]
```


DAMO Demo: Access-aware Program Profiling (1/2)

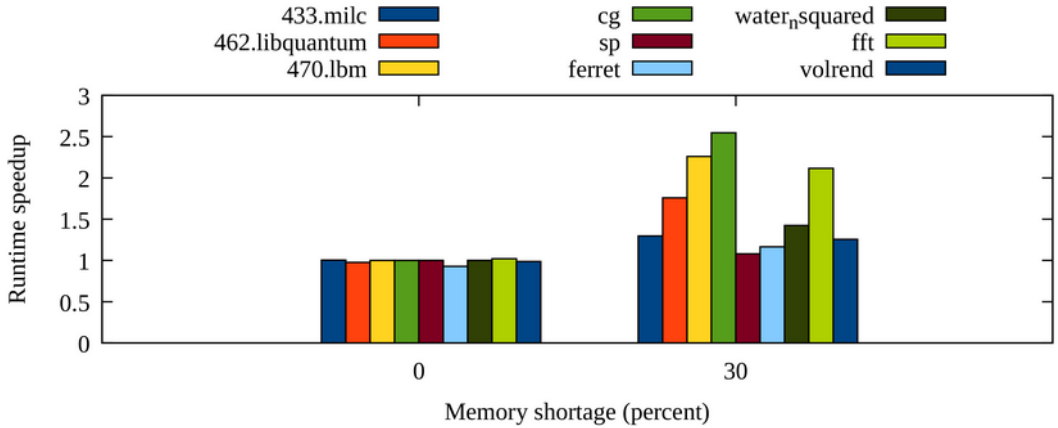
```
# damo report profile --access_rate 50% 100%
Samples: 75K of event 'cpu-clock:pppH', Event count (approx.): 18985500000
Overhead  Command          Shared Object         Symbol
  66.08%  swapper          [kernel.vmlinux]      [k] pv_native_safe_halt
  29.81%  fft              fft                   [.] Transpose
   0.18%  ps               [kernel.vmlinux]      [k] do_syscall_64
[...]
```

DAMO Demo: Access-aware Program Profiling (2/2)

```
# damo report profile --access_rate 50% 100%
Samples: 75K of event 'cpu-clock:pppH', Event count (approx.): 18985500000
Overhead  Command          Shared Object         Symbol
  66.08%  swapper          [kernel.vmlinux]      [k] pv_native_safe_halt
  29.81%  fft              fft                   [.] Transpose
   0.18%  ps               [kernel.vmlinux]      [k] do_syscall_64
[...]
# damo report profile --access_rate 0% 0%
Samples: 578K of event 'cpu-clock:pppH', Event count (approx.): 144693500000
Overhead  Command          Shared Object               Symbol
  66.50%  swapper          [kernel.vmlinux]            [k] pv_native_safe_halt
  12.81%  fft              fft                         [.] FFT1DOnce.constprop.1
  12.66%  fft              fft                         [.] Transpose
   0.95%  fft              [kernel.vmlinux]            [k] clear_page_orig
```

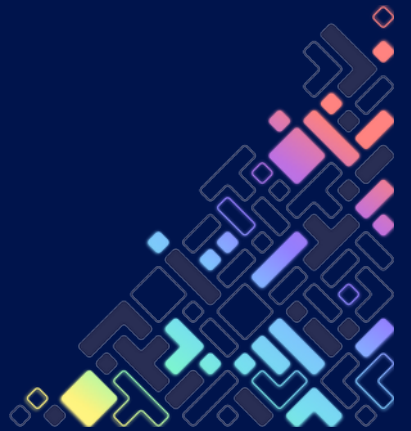
DAMO Demo: Manual Optimization of Program

- Optimize the program logic or add system calls such as `madvise()` or `mlock()`
- Achieved 2x performance improvement under memory pressure
- More details are out of the scope of this talk



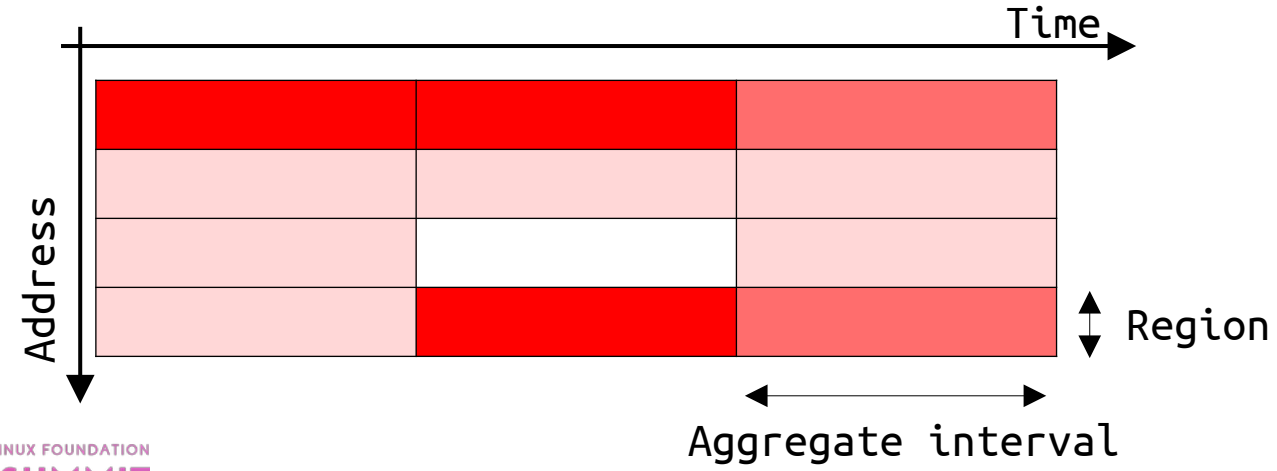
The image is retrieved from ksummit 19 DAMON talk

Online Access-aware System Operations



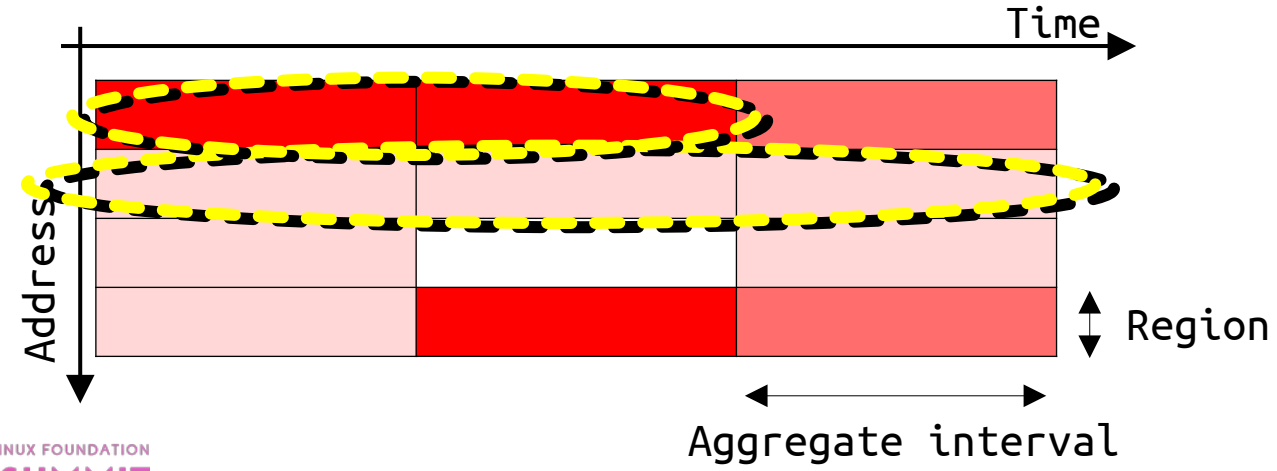
Inefficiency of Fixed Time Granularity Regions (1/2)

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



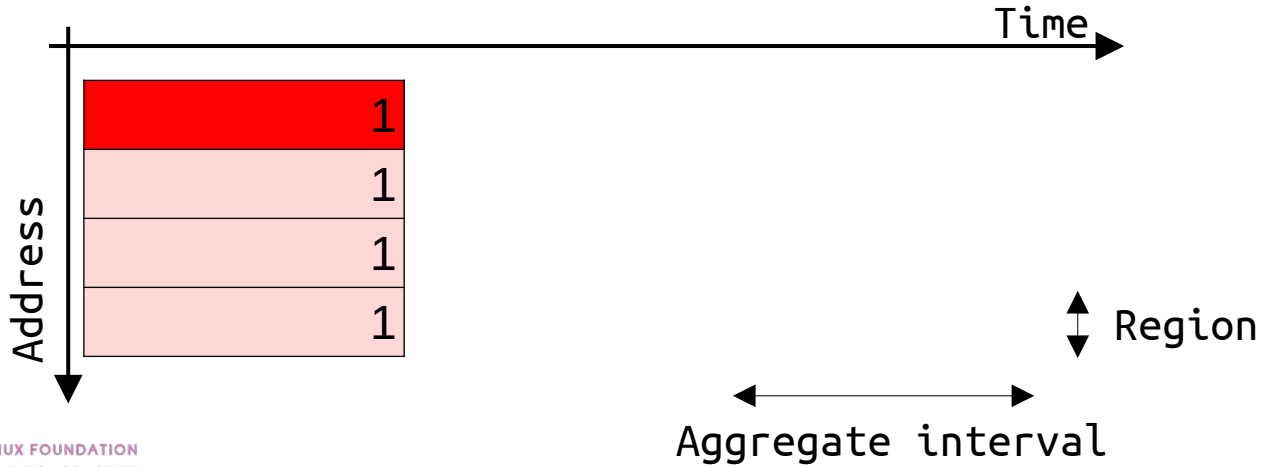
Inefficiency of Fixed Time Granularity Regions (2/2)

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



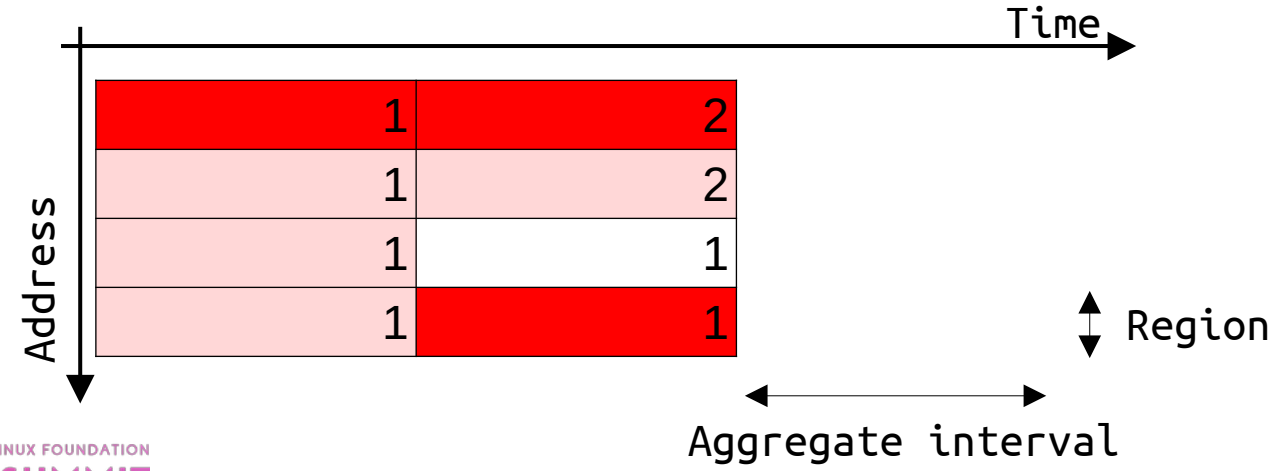
Dynamic Time Granularity (1/3)

- Count how long the hotness has kept
- Snapshot contains recent access history



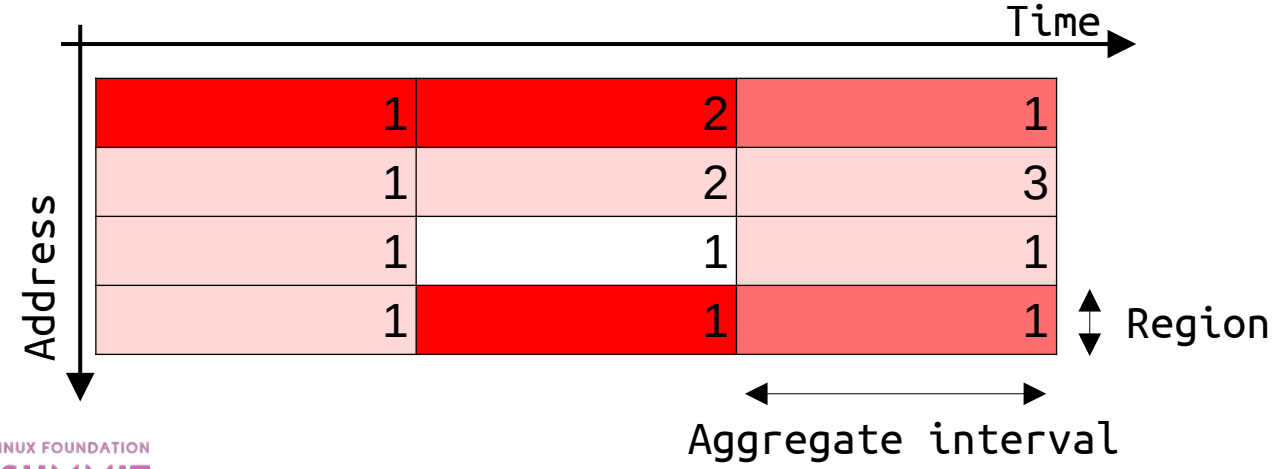
Dynamic Time Granularity (2/3)

- Count how long the hotness has kept
- Snapshot contains recent access history



Dynamic Time Granularity (3/3)

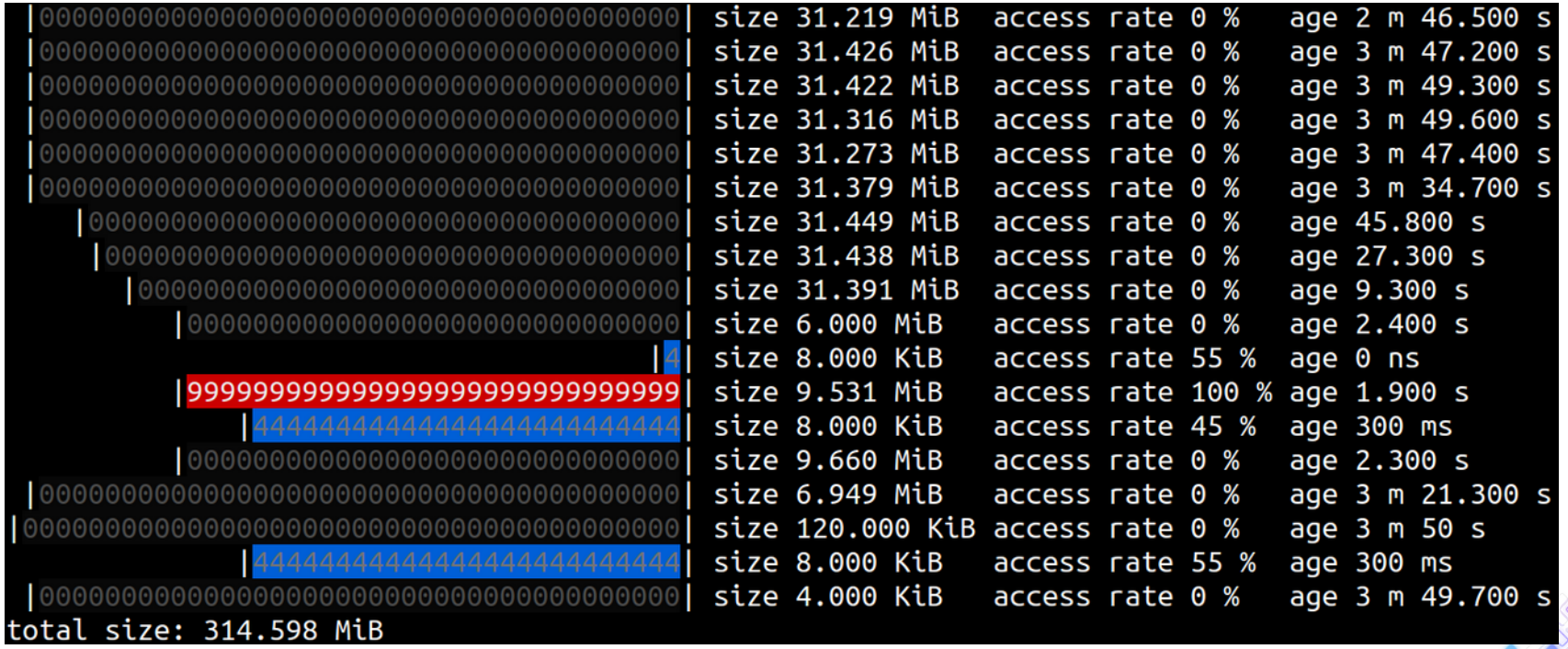
- Count how long the hotness has kept
- Snapshot contains recent access history



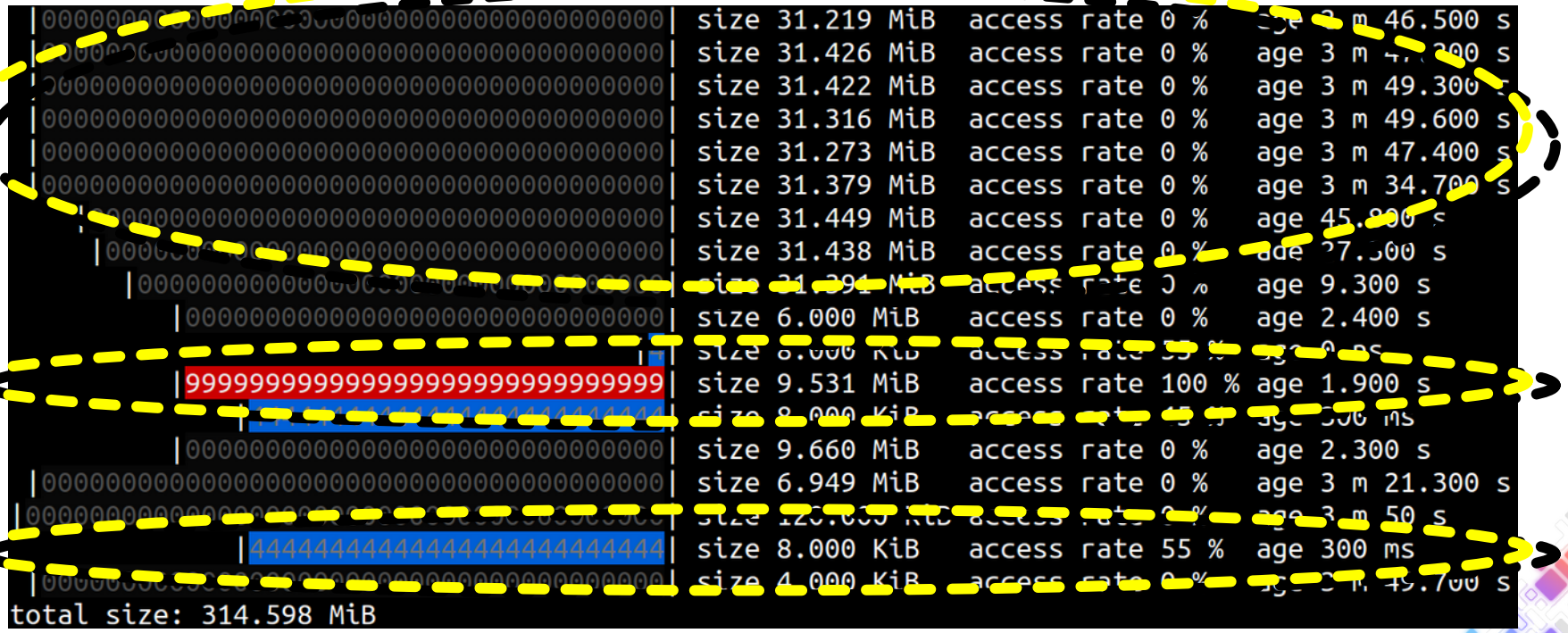
DAMO Demo: Showing Snapshot (1/3)

```
# damo start $workload_command
# damo show  --format_region \
        "<box> size <size> access rate <access rate> age <age>" \
        --region_box_min_max_height 1 1 --region_box_min_max_length 1 40 \
        --region_box_align right --region_box_colorset emotion
```


DAMO Demo: Showing Snapshot (2/3)

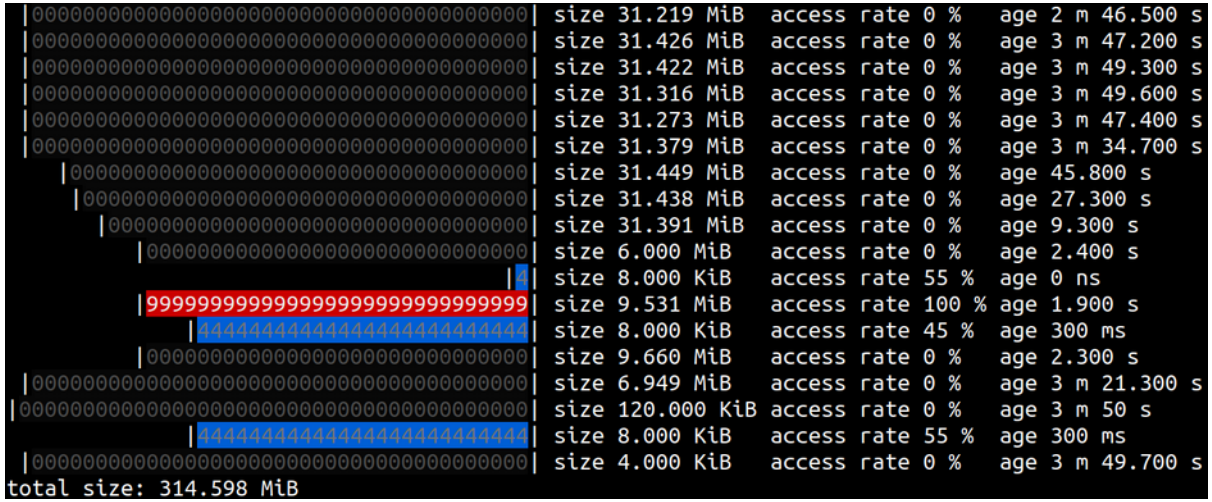


DAMO Demo: Showing Snapshot (3/3)



Snapshot: Real Output of DAMON

- Heatmaps: DAMO-collected snapshots
- Both time/space overheads are not ruled by memory size/monitoring time
- Informative enough for reasonable online optimizations



DAMOS: DAMon-based Operation Schemes

- A core feature of DAMON
- Apply specific operation to regions of specific access pattern
  - e.g., page out regions that not accessed for >=2 minutes
  - Get the latest snapshot from DAMON
  - Find regions of the pattern from the snapshot
  - Apply the action to the regions
  - Repeat
- Users specify the pattern and the action

DAMO Demo: Online Access-aware Optimization

- Page out regions that not accessed for >=5 seconds

```
# damo record "$HOME/parsec3_on_ubuntu/run.sh splash2x.fft" \
        --include_child_tasks --profile --footprint \
        --damos_action pageout --damos_access_rate 0% 0% --damos_age 5s max
[...]
real    2m 0.43s
[...]
# damo report footprints rss
[...]
# avr:  4.955 GiB
[...]
```

DAMOS Quotas: Safe-guards

- Finding optimum access pattern of the interest is challenging
- DAMOS resource usage quotas can be set for avoiding extreme misconfigs
- Under the limit, actions are applied to regions that prioritized for the action
- For pageout, colder memory regions are prioritized (paged out first)

DAMO Demo: DAMOS Quotas

- Page out regions that not accessed for >=5 seconds,
  up to 1GiB per second, using up to 100ms per second

```
# damo record "$HOME/parsec3_on_ubuntu/run.sh splash2x.fft" \
        --include_child_tasks --profile --footprint \
        --damos_action pageout --damos_access_rate 0% 0% --damos_age 5s max \
        --damos_quotas 100ms 1G 1s
[...]
51.772s
[...]
# damo report footprints rss
[...]
# avr:  8.527 GiB
[...]
```

DAMOS Quota Goals: Aggressiveness Auto-tuning

- Knowing the optimum quota is still difficult; policy is coupled with mechanism
- Users specify aiming system metric values
- DAMOS self-tune its quota to achieve the goal using a simple feedback loop

DAMO Demo: DAMOS Quota Goals

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

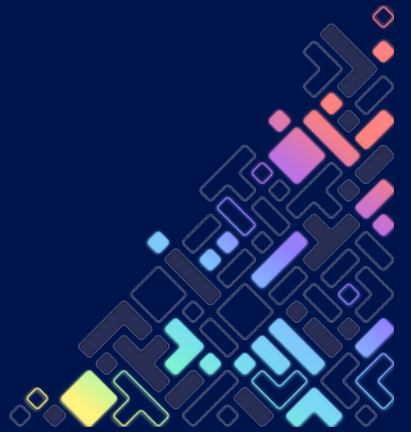
```
# damo record "$HOME/parsec3_on_ubuntu/run.sh splash2x.fft" \
        --include_child_tasks --profile --footprint \
        --damos_action pageout --damos_access_rate 0% 0% --damos_age 5s max \
        --damos_quotas 100ms 1G 1s \
        --damos_quota_goal some_mem_psi_us 10000
[...]
real    49.741s
[...]
# damo report footprints rss
[...]
# avr:  9.721 GiB
```

DAMOS Results At a Glance

- DAMOS improves system efficiency
- Quota: Safe-guard
- Quota goal: Aim-driven self-driver
- Daily DAMOS performance test results are available online

	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

DAMON Community



DAMON Is Developed-By: The Community

- Jie Li et al. wrote EuroSys'24 paper exploring DAMON in depth
- Intel wrote a paper and posted patches for DAMON accuracy improvement
- SK Hynix developed their tiered memory management kit using DAMON; collaborating with the community on upstreaming
- A number of DAMON features are results of community discussions

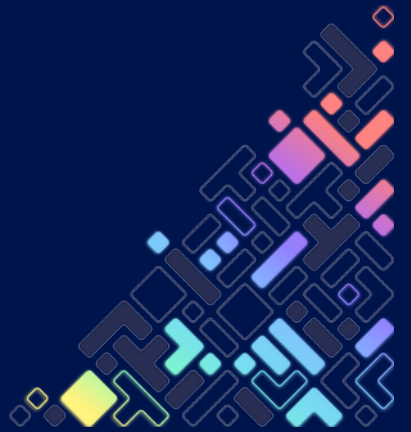
Communication Channels

- DAMON-dedicated open mailing list: https://lore.kernel.org/damon/
- Bi-weekly community meetup series
  - An in-person version of it will be held *Today* 4:55 pm, Room 444
- Having occasional/regular private meetings on demand

Call For Your Voices

- We prefer random evolution over intellectual design
- Put your voice on the evolution path for your purpose
  - Report your use case/test results and challenges
  - Ask questions and request features
  - Show your interest to known future works
  - Send patches

Summary



Summary

- Access-aware system operation is required for modern memory systems
- DAMON: best-effort access monitoring kernel subsystem
- DAMOS: Access-aware system operation engine
- DAMO: User-space tool for DAMON/DAMOS
- DAMON is evolving under the community’s efforts

Questions?

- If your question is not answered by the session, use below
  - The maintainer: **sj@kernel.org**
  - Project webpage: **https://damonitor.github.io**
  - Kernel docs for **admin** and **programmers**
  - DAMON mailing list: **damon@lists.linux.dev**
  - DAMON Beer/Coffee/Tea **Chat**
  - Today’s in-person DAMON **meetup** (4:55 pm, room 444)