LinuxCon @ Open Source Summit North America
Vancouver, Canada, 2023-05-12

DAMON, DAMOS, and DAMO:

Kernel Subsystems and User-Space Tools for Data Access-Aware System Analysis/Optimizations

SeongJae Park <sj@kernel.org>

https://damonitor.github.io

I, SeongJae Park (sjpark@)

- Call me SJ, or whatever better for you to pronounce
- Working on AWS
- Maintainer of DAMON

Overview

- Why Access-awareness Matters
- DAMON: Data Access MONitor
- DAMOS: DAMON-based Operation Schemes
- DAMO: DAMon Operator
- Putting It Altogether: Access-aware Linux (AL) System
- Collaborations and Use Cases
- Conclusion
- QnA

Why Access-awareness Matters

Modern Systems Are Using Hierarchical Memory

- There are memory devices having different characteristics
  - Devices: Registers, Cache, DRAM, Flash, Disk, Tape, ...
  - Characteristics: Capacity, Latency, Bandwidth, Power efficiency, Cost, ...
  - Faster ones tend to be expensive and power-consuming
- Modern systems use those in a hierarchical manner for efficiency
  - L1$ on top, L2$ next, L3$, DRAM, SSD, HDD, Tape, so on
- The hierarchy will only becomes more complicated with new devices
  - Zram-like software-defined memory between DRAM and SSD
  - CXL-Memory-like devices between DRAM and SSD
  - Fabric-based in-rack devices between somewhere
  - Network-based devices somewhere

Modern Systems Are Using Hierarchical Memory

- The hierarchy will only becomes more complicated

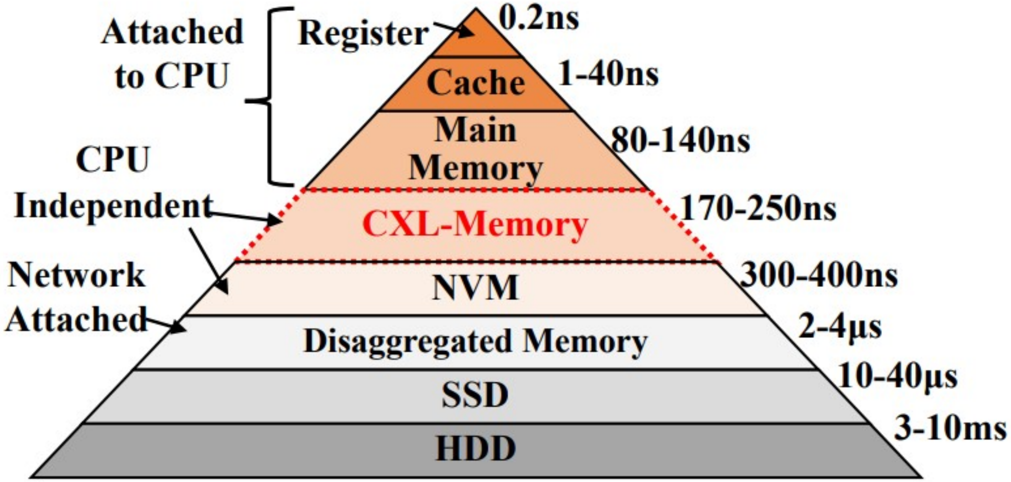


Figure 2: Latency characteristics of memory technologies.

https://dl.acm.org/doi/pdf/10.1145/3582016.3582063

Cost and Importance of Memory is Increasing

- Modern workloads are becoming more and more data intensive
  - Machine learning, cloud, bigdata, …
- DRAM Price is not dramatically dropped
- Meta reports high cost of DRAM on their data centers
  - 33.3% of cost and 37.1% of power consumption

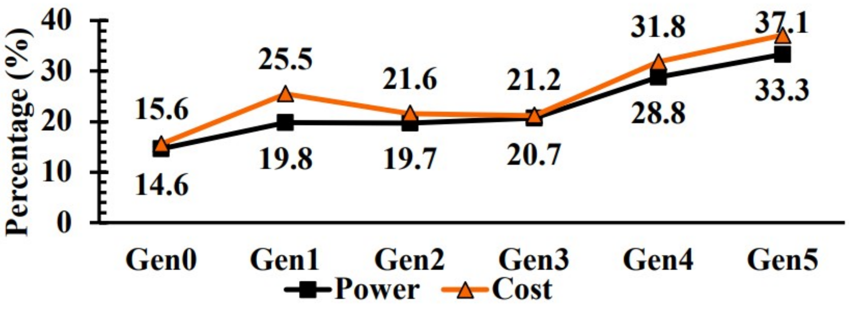


Figure 3: Memory as a percentage of rack TCO and power across different hardware generations of Meta.

https://dl.acm.org/doi/pdf/10.1145/3582016.3582063

Access-awareness is Critical for Efficiency

- Given the trend, efficient memory management is critical for the cost
  - “The year of efficiency” might not end by 2023
- Efficient memory management would be simple
  - Keep important data close, keep critical data closer to fastest memory device
- What data is important?
  - Data that will (or assumed to) be accessed frequently in near future
- How to know those?
  - Fortune-teller or ChatGPT could be irresponsible, better to make data-driven decisions
  - Monitor current data access pattern and predict future

Examples of Access-aware Efficiency Optimizations

- Access-aware Trasnaprent Huge Pages (THP) collapse/split
  - Using huge pages increase performance by reducing TLB misses
  - But also increase memory footprint due to huge page internal fragmentation
  - Using THP for only hot data reduces the memory footprint while preserving the performance
  - The work was accepted to a top-tier conf (OSDI)
- Proactive reclamation
  - Reactive memory reclamation under memory pressure (Linux kernel’s default behavior) incurs latency spikes and keep unnecessarily big memory footprints
  - Proactively finding and reclaiming cold pages reduce such spikes and reduce memory footprint
  - The works from Google and Meta were accepted to another top-tier conf (ASPLOS) twice, and being used on their fleets
- Ideas sound simple, but some ~~DAMON~~ demon is in the detail

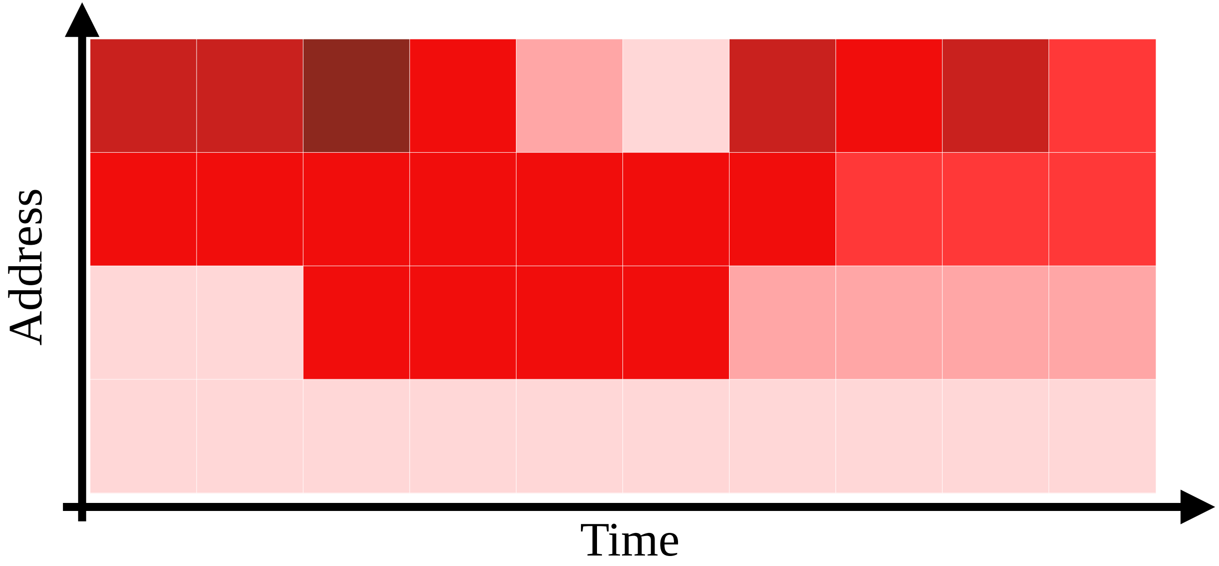
DAMON: Data Access MONitor

Monitoring Data Accesses Is Not an Easy Problem

- With prior-DAMON normal access monitoring techniques,
  - Overhead is high, and arbitrarily grows as the size of memory to monitor increases
  - High overhead can also affect monitoring results accuracy
    - Maybe uncertainty principle can be applied here
- Linux kernel made some tradeoffs for low overhead
  - Called LRU, but not real LRU (even real LRU might not be the ideal)
  - No fine-grained access tracking but hotness classification
    - traditionally only accessed at least once or not, active or inactive
    - MGLRU improved this with generation concept
  - Access scan happens reactive to memory pressure-like events
    - Without periodic and frequent such events, the accuracy could be bad
  - Worked well so far, but could be more challenging, given the trend

DAMON: What It Provides?

- Conceptually, DAMON does periodic access check of all memory area
- Inform users when (x-axis) which memory (y-axis) area has how frequently accessed (color)



DAMON: What Is Special?

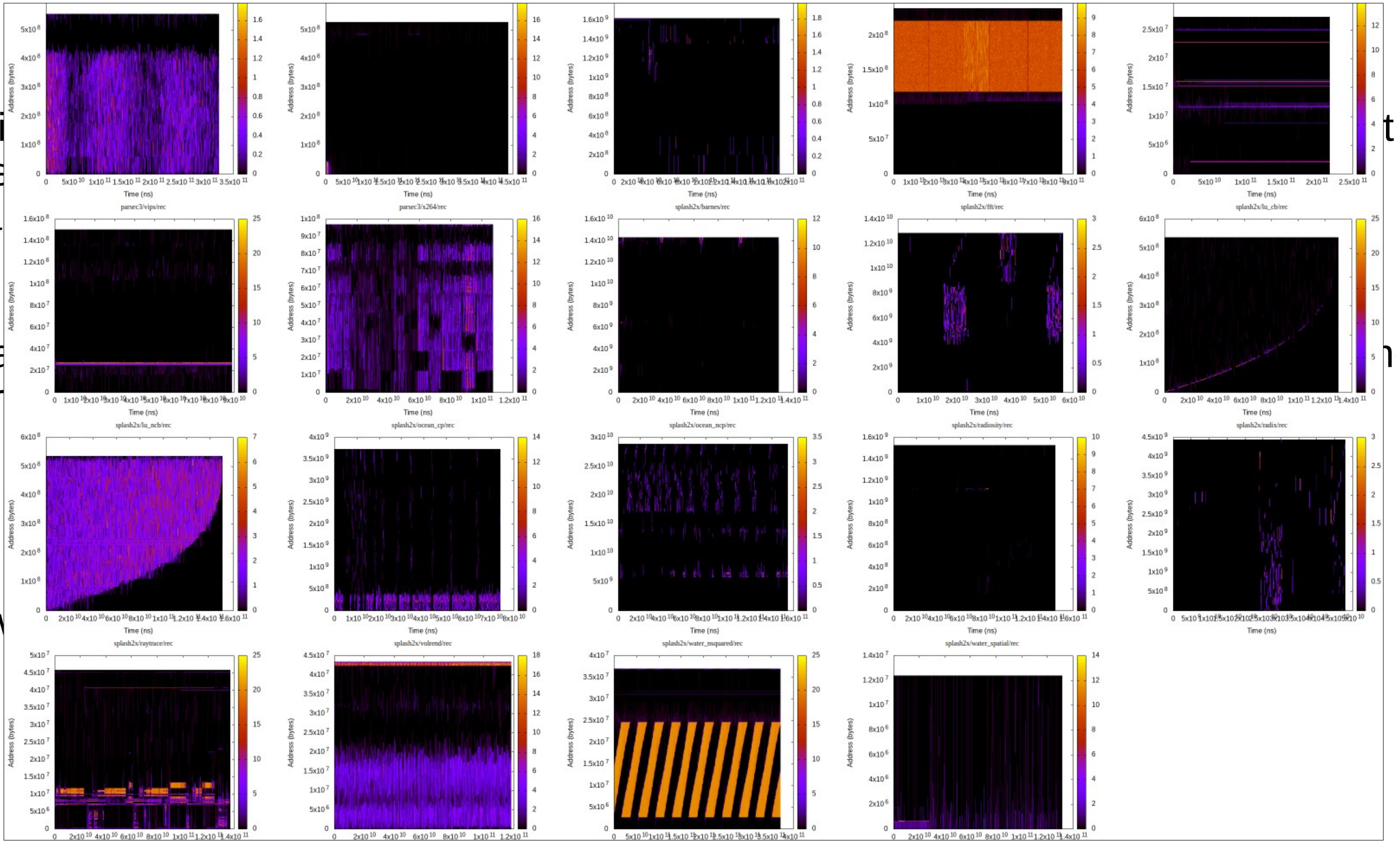
- Equips a simple but effective best-effort overhead-accuracy tradeoff logic
  - Called “regions-based sampling” and “adaptive regions adjustment”
  - Let users set the min accuracy and max overhead
  - make best-effort for lowest overhead and max accuracy under the constraints
  - For detail, please read the doc or the paper
  - Important parts of DAMON, but not itself; Could be replaced or unused if needed, in a future
- Not a magic, just a trade-off
  - May provide a poor result if wrongly tuned
  - How well the best-effort works? Only data can say

Evaluation of DAMON’s Overhead and Accuracy

- DAMON is lightweight
  - A prior-DAMON page-granularity approach (kstaled) consumes 100% single CPU time to scan accesses to 512GB memory every 2 minutes
    (scans 512GB / (2mins * 100% CPU time) = 4.26 GB/s)
    - There were great improvements following the first approach, though
  - On a production setup, DAMON _*conceptually*_ scans accesses to 68.60 GB memory every 5 msecs with <1% single CPU time (scans 68.6GB / (5ms * 1% CPU time) = 1,372 TB/s)
    - Note: DAMON upper-bound overhead can be set regardless of the memory size
- DAMON is accurate
  - Shows sane monitoring results with realistic benchmarks

Evaluation of DAMON’s Overhead and Accuracy

- DAMON
  - A pri … time to scan acce … (sca
    -
  - On a … nory ever … 1,372 TB/s)
    -
- DAMON
  - Show



Evaluation of DAMON’s Overhead and Accuracy

- DAMON is lightweight
  - A prior-DAMON page-granularity approach (kstaled) consumes 100% single CPU time to scan accesses to 512GB memory every 2 minutes
    (scans 512GB / (2mins * 100% CPU time) = 4.26 GB/s)
    - There were great improvements following the first approach, though
  - On a production setup, DAMON _*conceptually*_ scans accesses to 68.60 GB memory every 5 msecs with <1% single CPU time (scans 68.6GB / (5ms * 1% CPU time) = 1,372 TB/s)
    - Note: DAMON upper-bound overhead can be set regardless of the memory size
- DAMON is accurate
  - Shows sane monitoring results with realistic benchmarks and a production setup
    - found 7GB working set and 4KB hottest region from the production setup
  - Identifying hot memory regions from DAMON results with human eyes and modifying the program to do `mlock()` the regions achieves up to 2.55x speedup under memory pressure[1,2]

DAMON Demonstration

- https://sjp38.github.io/post/damon/#demo-video

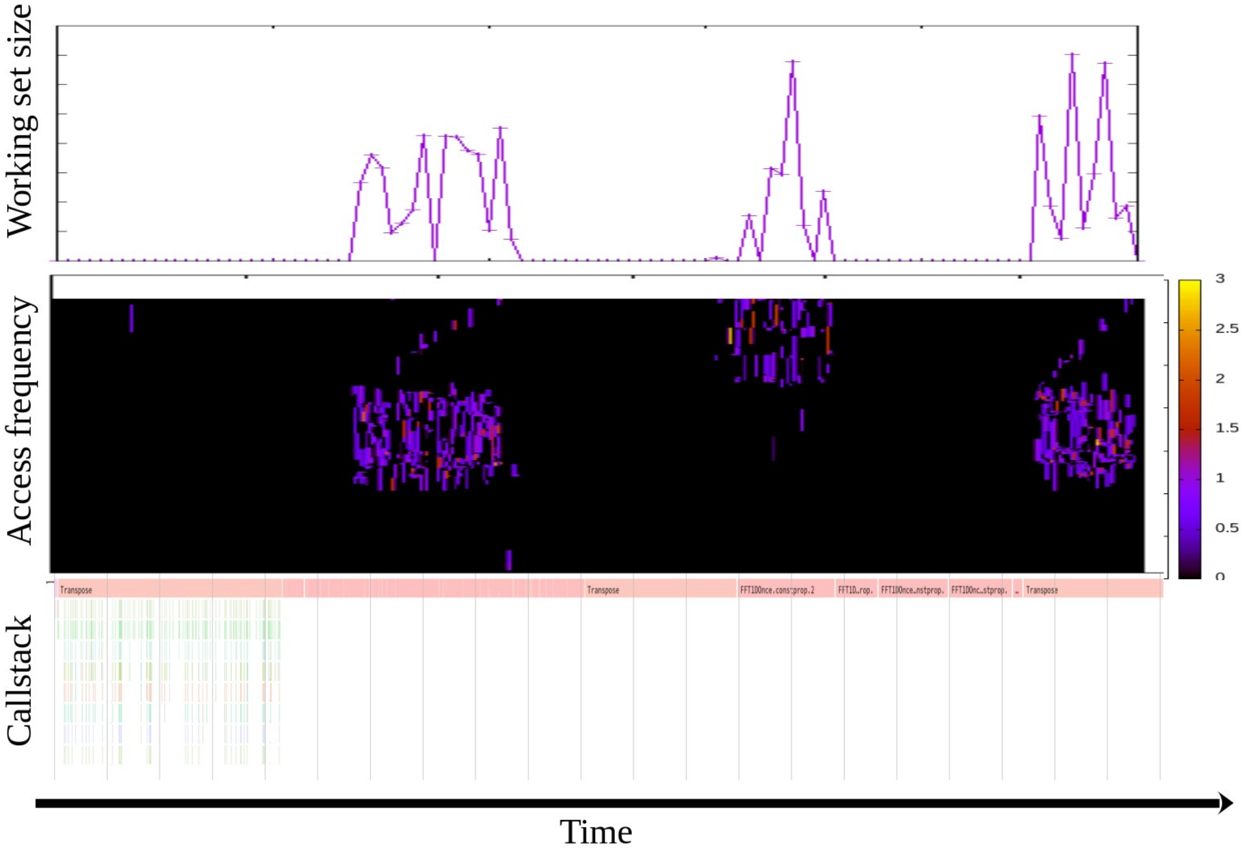
```
# while :; do ps -o rss=,cmd= --pid $(pidof masim); sleep 1; done # screen 1
# while :; do ps -o rss=,cmd= --pid $(pidof kdamond.0); sleep 1; done # screen 2
# damo monitor “./masim configs/stairs.cfg”
# ./masim -repeat 1024 configs/stairs.cfg
# damo record $(pidof masim)
# damo report raw
# damo report heats -heatmap
# damo report wss
```

DAMON Allows Access-aware System Operations

- A number of access-aware analysis and optimizations of systems are possible
  - Providing monitored working set size based system config guideline
  - Debugging/optimizing wrongly implemented data access logics

Example DAMON-based Profiling

- https://sjp38.github.io/post/damon_profile_callstack_example/



DAMON Allows Access-aware System Operations

- A number of access-aware analysis and optimizations of systems are possible
  - Providing monitored working set size based system config guideline
  - Debugging/optimizing wrongly implemented data access logics
  - System level access-aware memory management
    - This is something Linux kernel can further help

DAMOS:
DAMON-based Operation Schemes

Yet More Things DAMON Can Help

- Common DAMON-based optimizations would be…
  - Get/analyze monitoring results and [de]prioritize memory regions based on the analysis (e.g., page out cold regions, apply THP to hot regions, …)
  - IOW, getting important data close, critical data closer
- May contain repetitive and inefficient steps
  - The analysis and prioritization could be repetitive
  - User-space driven management could be inefficient
    - Transfer monitoring results from kernel to user space
    - Transfer management decision from user to kernel space
- Can’t DAMON do that instead, directly inside the kernel?

DAMOS for Access-aware Optimizations with No Code

- DAMOS is a feature of DAMON for offloading the optimizations effort to DAMON
  - Users can simply
    - specify the access pattern of their interest, and
    - the action they want to apply to the regions of the pattern
  - Then, DAMON finds regions of the pattern and apply the action
  - No code, just request specification

```
$ damo start --schemes “
[{
        "access_pattern": {
                "nr_accesses": {"max": "0 %"},
                "age": {"min": "2 m"}
        },
        "action": "pageout"
}]”
```

A json-format DAMOS scheme asking
“Page out memory regions of >=4K that not accessed at all for >=2 minutes”

Evaluation of DAMOS’ Effectiveness

- Implemented main ideas of the two state-of-the-arts works with DAMOS
  - Access-aware THP collapse/split
    - Our version removes 76.15% of THP memory bloats
      while preserving 51.25% of THP speedup
  - Proactive reclamation
    - reduces 93.38% of residential sets and 23.63% of system memory footprint
      while incurring only 1.22% runtime overhead in the best case.
  - For more details, please read the report
- Reasonable DAMOS effectiveness also means reasonable DAMON accuracy

DAMOS Demonstration

- https://sjp38.github.io/post/damon/#demo-video

```
# while :; do ps -o rss=,cmd= --pid $(pidof masim) ; sleep 1; done # screen 1
# while :; do ps -o rss=,cmd= --pid $(pidof kdamond.0); sleep 1; done # screen 2
# ./masim -repeat 1024 configs/stairs.cfg
# damo start --damos_access_rate 0% 0% --damos_age 5s max \
             --damos_action page $(pidof masim)
```

You may show rss of ‘masim’ process is significantly reduced;
DAMON worker thread (kdamond.0) may show a little higher CPU usage for pageout

DAMOS Feature for Production: Quotas

- Schemes’ target access patterns should fine tuned to be appropriately aggressive
  - If too aggressive, DAMON overhead for applying the action could be significant
  - If not aggressive enough, no effective change will be made
  - Finding the optimal value might be doable for big companies, but DAMON is for all
- DAMOS Quotas allow users set upper-limits of the aggressiveness in an intuitive way
  - Limit time for and bytes to apply the action per specific time
  - Under the limit, prioritize regions using their access pattern
    - Allow users set their personal priority weights for each access pattern element

```
"quotas": {
    "time_ms": "10 ms", "sz_bytes": "100 MB", "reset_interval_ms": "1 s",
    "weights": {
        "sz_permil": "0 %", "nr_accesses_permil": "50 %", "age_permil": "50 %"
    }
},
```

“Apply the scheme under 10ms and 100 MB per second limit,
treating access frequency and age of regions samely important, while ignoring sizes of the regions”

DAMOS Feature for Finer User Control: Filters

- Some users may know some characteristics of their workloads more than kernel
  - E.g., list of latency-critical processes, frequent anonymous page usage, …
- DAMOS Filters allows users to filter schemes via type of the pages
  - Currently support anonymousness and belonging cgroups of pages
  - Users can apply a scheme to only [non-]anon pages, pages of specific cgroups, in any combination

```
"filters": [
    {"filter_type": "anon", "matching": true}
    {"filter_type": "memcg", "memcg_path": "/latency-critical", "matching": true},
]
```

“Do not apply the scheme to anonymous pages of ‘latency-critical’ cgroup”

Is DAMON/DAMOS for User-space Control? Or Kernel That Just Works?

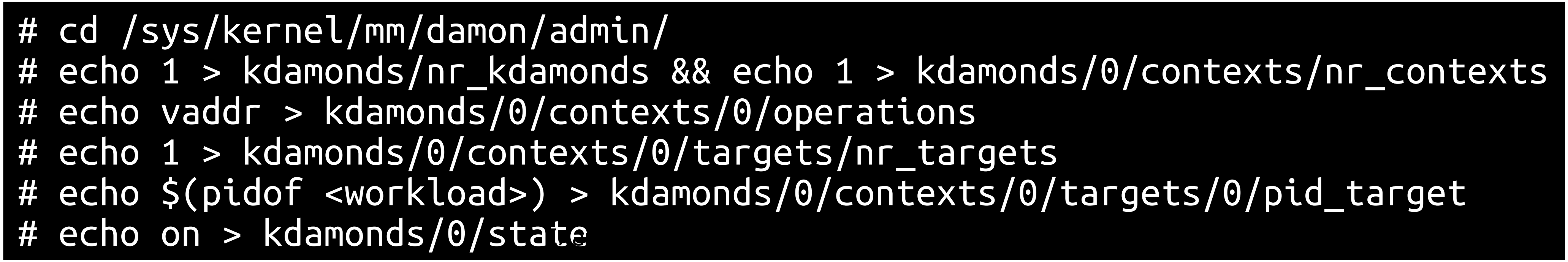
- Users having capacity could get more information, and need a way for finer control
  - DAMOS Filters could be used for them
- Users having no such capacity need a kernel that just works
  - DAMOS Quotas could be used for them
- We’re pursuing to help both parties
  - More features for both parties will be developed

DAMO:
Data Access Moniotring Operator

(User-space Tool written in Python)

DAMON User Interfaces: How You Can Use DAMON

- DAMON provides only kernel API for other kernel components
- There is a Linux kernel module named DAMON sysfs interface
  - Create pseudo-files on sysfs, hook I/O to the files
  - As a response to the I/O, control DAMON using DAMON API
  - Resulting in DAMON ABI, that user-space can use
  - Manual use of the files by human fingers is tedious and buggy, so discouraged
  - Easy-to-use User-space tools can be developed using the ABI



```
# cd /sys/kernel/mm/damon/admin/
# echo 1 > kdamonds/nr_kdamonds && echo 1 > kdamonds/0/contexts/nr_contexts
# echo vaddr > kdamonds/0/contexts/0/operations
# echo 1 > kdamonds/0/contexts/0/targets/nr_targets
# echo $(pidof <workload>) > kdamonds/0/contexts/0/targets/0/pid_target
# echo on > kdamonds/0/state
```

DAMO: Data Access Monitoring Operator

- Human-friendly user-space tool for DAMON
  - Developed by DAMON author (maybe we could call this an official tool)
  - Provides human-friendly user interface for input access pattern visualization
  - Written in Python and available at PyPI
- DAMO is not necessarily the only one DAMON user-space tool
  - Anyone can write their own DAMON user-space tools using the ABI
    - DATOP: developed by Alibaba, available at Github

DAMO as a Library for Your Own DAMON User-space Tool

- _damon.py implements core logics for DAMON ABI
- DAMO commands are implemented by using _damon.py as a library
  - Refer to damo_record.py, damo_start.py, and damo_stop.py for example usage
- You could implement DAMON user-space tool for you using it as a library
  - The interface is not that stable at the moment, though
    - Best approach might be making your tool merged in DAMO and maintaining it on your own
    - Or, use it as only a reference implementation

DAMO Demo

- https://sjp38.github.io/post/damon/#demo-video
- Wait, you already shown the demonstration!

Putting It Altogether:

Access-aware Linux (AL) System

Putting It Altogether: Access-aware Linux (AL) System

- DAMON: Data Access MONitor
  - Provides practical best-effort quality information with low overhead
- DAMOS: DAMON-based Operation Schemes
  - Allows common access-aware memory management with no code
- DAMO: Data Access Monitoring Operator
  - Provides human-friendly user interface for DAMON and DAMOS
- Those construct an Access-aware Linux System
  - Allow DRAM cost cut and memory intensive performance boosting
  - Not an official but a temporal name for making Amazon Linux (AL) people be confused

Availability of Access-aware Linux

- All AL system components are open source and upstreamed
  - DAMON and DAMOS have merged in Linux mainline v5.15 and v5.16, respectively
  - DAMO is available at PyPi
    - Might be better to be in the Linux mainline?
- There are options for people who cannot use >=5.15 kernels
  - Basic DAMON/DAMOS functionalities are backported on Amazon Linux 5.4 kernel
  - All new DAMON/DAMOS features are being backported to Amazon Linux 2 5.10 kernel
  - Quite amount of DAMON/DAMOS features are backported on Android 5.10 common kernel
  - There’s a rumor about some customers’ DAMON backporting request to a distro

Use-cases and Collaborations

Use-cases

- Disclaimer: These are just rumors and clues collected from private/public conversations
  - There’s no central and well-managed channel and storage for this kind of information
  - There could be a lot of false positives/negatives
- Some people from companies including Alibaba, AMD, AWS, DigitalOcean, Google, Huaweii, IBM, Meta, Oppo, Oracle, etc seems using or experimenting it
  - Android common kernel ported and enabled DAMON_RECLAIM
- Some academic/industry folks are researching DAMON-based tiered memory management

Collaborations

- Collaborating with a number of AWS internal/external people (DAMON community)
- In 2022, 39 Amazon-external people contributed 83 patches for DAMON
- Communicating in several ways
  - DAMON-dedicated development mailing list
  - Virtual bi-weekly community meetup series
  - Presenting DAMON in conferences since 2019
    - Striving to do those for both kernel and user space application developers
  - Having occasional/regular private meetings on demand

DAMON Community is Waiting For Your Voices

- DMAON, DAMOS, and DAMO are still under active development; the journey just begun
  - The interface may not fit for you
  - There could be lacking features for you
  - Not meaning unstable; will always be backward compatible and never breaks users
- Don’t forgive it or wait for someone to implement it; make your voice
  - Report your use case, challenges, and benefits you currently getting
  - Ask questions and request features for your use case
  - Prioritize some future works by
    - Showing your interest, share your expected usage of the feature
    - Test and share the results
  - Send patches

Conclusion

- DAMON: Linux kernel subsystem for efficient data access monitoring
- DAMOS: No-code efficient access-aware system optimization engine
- DAMO: User-space tool/library for DAMON and DAMOS
- Some people are getting fun with those
- You can help delivering more fun to the world with those

Questions?

- You can also use
  - 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**

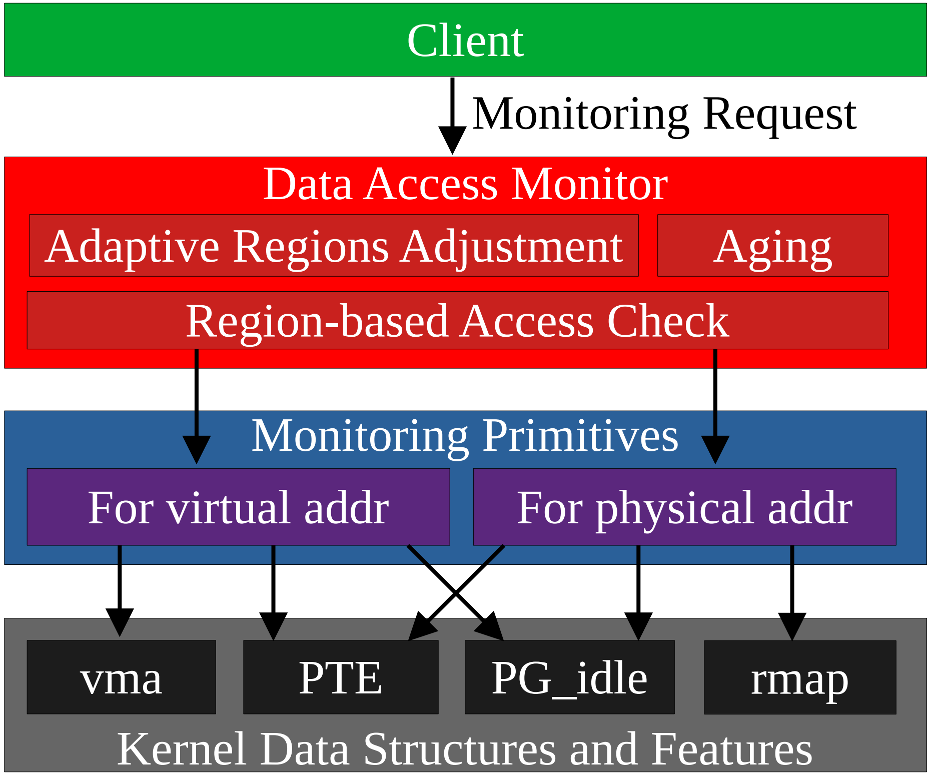
Backup Slides

Conceptual Psudo-code of DAMON

```
while monitoring_on:
    for page in monitoring_target:
        if accessed(page):
            nr_accesses[page] += 1
    if time() % aggregation_interval == 0:
        for callback in user_registered_callbacks:
            callback(monitoring_target, nr_accesses)
        for page in monitoring_target:
            nr_accesses[page] = 0
    sleep(sampling interval)
```

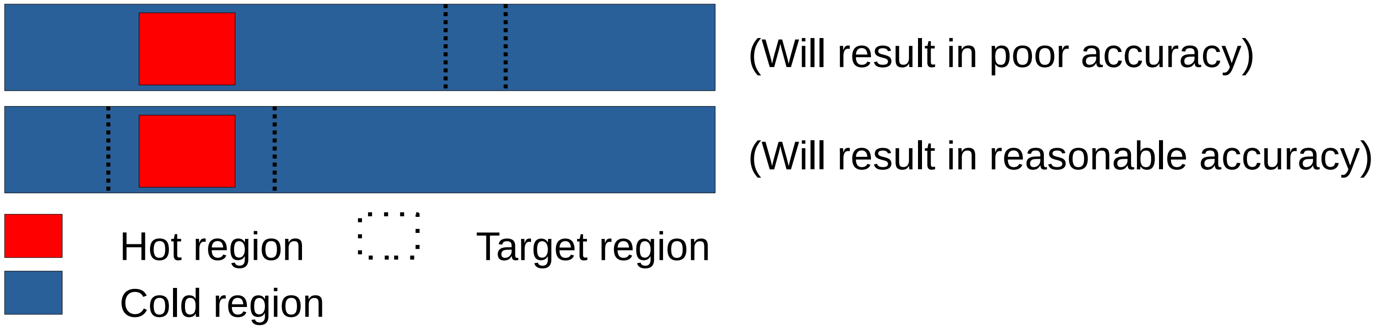
DAMON: Resulting Architecture

- Core logic and monitoring operations layer are separated
  - Multiple address spaces and usages can easily supported



Region-based Sampling

- Defines data objects in access pattern oriented way
  - “A data object is a contiguous memory region that all page frames in the region have similar access frequencies”
  - By the definition, if a page in a region is accessed, other pages of the region has probably accessed, and vice versa
  - Thus, checks for the other pages can be skipped
- By limiting the number of regions, we can control the monitoring overhead regardless of the target size
- However, the accuracy will degrade if the regions are not properly set



(Will result in poor accuracy)

(Will result in reasonable accuracy)

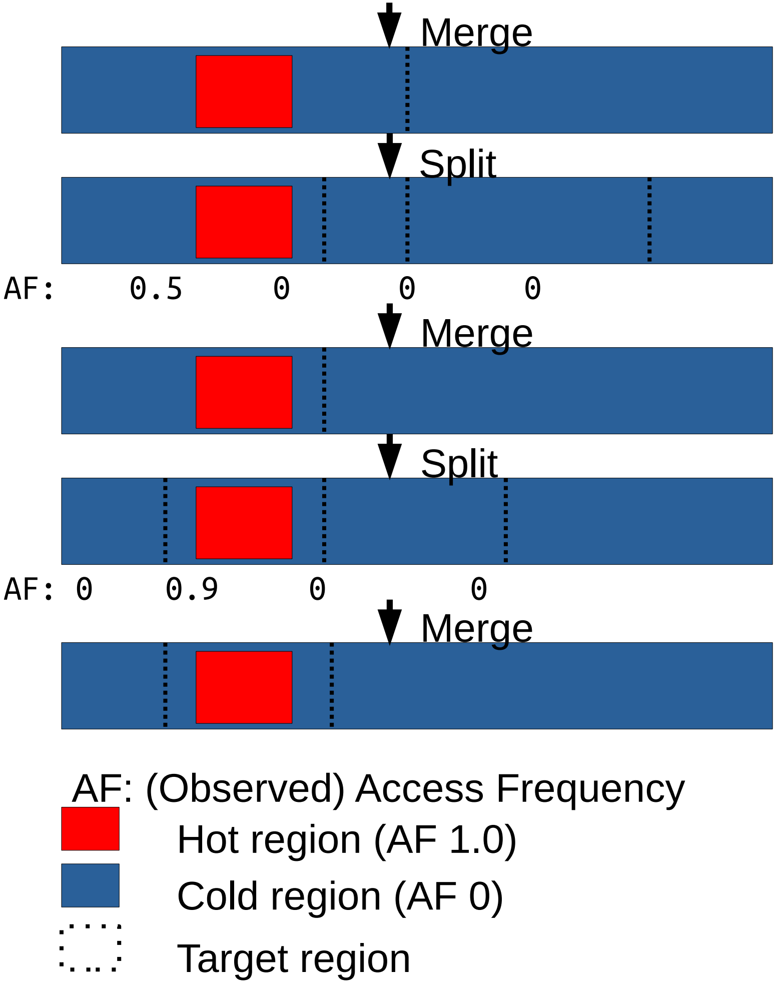
🟥 Hot region

⬚ Target region

🟦 Cold region

Adaptive Regions Adjustment

- Starts with minimum number of regions covering entire target memory areas
- For each aggregation interval,
  - merges adjacent regions having similar access frequencies to one region
  - Splits each region into two (or three, depend on state) randomly sized smaller regions
  - Avoid merge/split if the number of regions might be out of the user-defined range
- If a split was meaningless, next merge process will revert it (vice versa)
- In this way, we can let users control the upper bound overhead while preserving minimum and best-effort accuracy



Merge
Split
AF: 0.5 0 0 0
Merge
Split
AF: 0 0.9 0 0
Merge
AF: (Observed) Access Frequency
Hot region (AF 1.0)
Cold region (AF 0)
Target region