LinuxCon @ Open Source Summit Europe
Bilbao, Spain, 2023-09-19

Data Access Monitoring Operator (DAMO): User-Space Tool/Python Library for Access-Aware Profiling and Optimization of Your Linux Systems

SeongJae Park <sj@kernel.org>

https://damonitor.github.io

Slides are also available at: https://github.com/damonitor/talks/tree/master/2023/ossummit_eu

Notices

- The views expressed herein are those of the speaker;
  they do not reflect the views of his employers
- The talk is based on DAMO v1.9.9 and Linux kernel v6.6-rc2
- Some details can be changed in future versions of those

https://twitter.com/sjpark0x00/status/1295387149018300419/photo/1

I, SeongJae Park (sjpark@)

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

I, SeongJae Park (sjpark@)

- Call me SJ while speaking in English
- Working on AWS
- Maintaining Linux kernel DAMON subsystem and its user-space tool, DAMO
- A certified pilgrim (walked about a 500km preparing this talk)

Overview

- Why Access-awareness Matters
- DAMO: Data Access Monitoring Operator
- DAMO Inputs for Data Access Monitoring
- DAMO Inputs for Access Monitoring-based System Operations
- DAMO~~N~~ Community: Call For Participate
- Conclusion
- QnA

Why Access-awareness Matters

Hierarchical Memory

- Memory devices of different characteristics
  - Devices: Registers, Cache, DRAM, Flash, Disk, Tape, etc
  - Characteristics: Capacity, Latency, Bandwidth, Power efficiency, Price, etc
  - Faster ones tend to be small, expensive, and power-consuming (emits heat and CO2!)
- Hierarchical combination of the devices for efficiency
  - L1$ on top, L2$ next, L3$, DRAM, SSD, HDD, Tape, so on
- More peculiar memory devices for more complicated but efficient hierarchy are upcoming
  - Z{ram,swap}-like S/W-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

Hierarchical Memory

- Memory devices of different characteristics
  - Devices: Registers, Cache, DRAM, Flash, Disk, Tape, ...
  - Chara
  - Faste 2!)
- Hierarchi
  - L1$ c
- More pec
  - Z{ran
  - CXL-
  - Fabri
  - Network-based devices somewhere

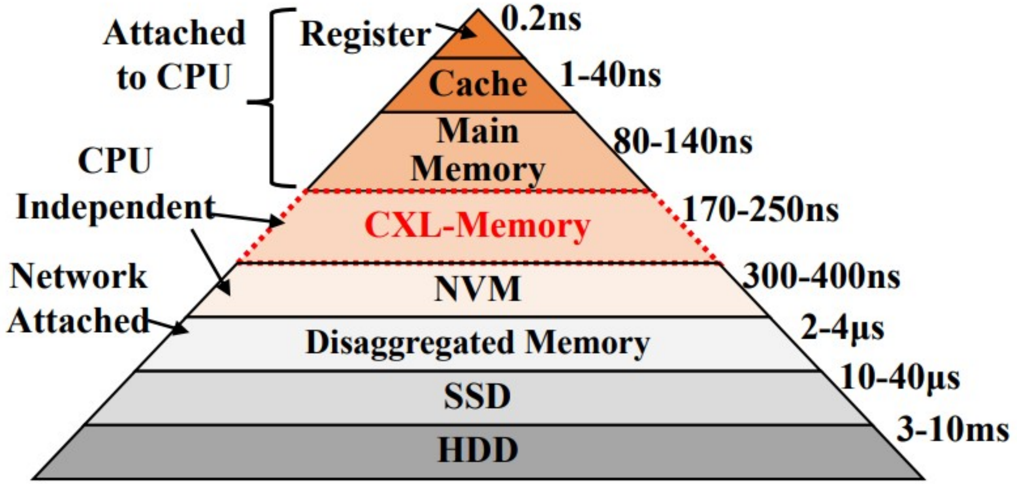


Figure 2: Latency characteristics of memory technologies.

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

Trend of Cost and Importance of Memory

- Modern workloads continuously becoming more data intensive
- DRAM price has not dramatically dropped
- Meta reports high cost of DRAM on their data centers
  - 33.3% of cost and 37.1% of power consumption for memory
- Cost and importance of memory system is significant, and expected to further increase

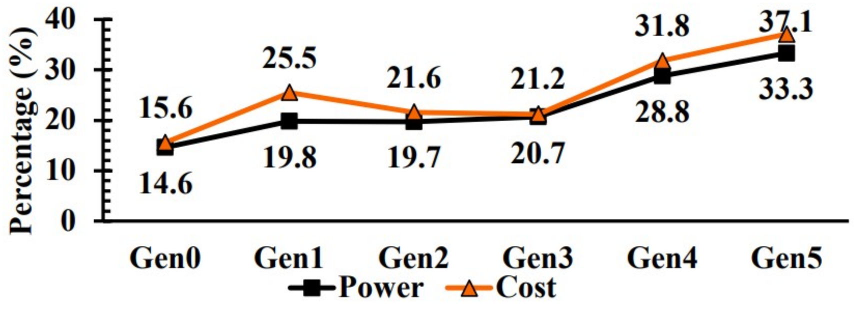


Figure 3: Memory as a percentage of rack TCO and power across different hardware generations of Meta.

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

Access-aware Efficient System Operations Needed

- Given the trends, hierarchical memory system operation is critical for efficiency
- Efficient access-aware memory system operation idea: Simple
  - Keep important data items close, keep critical data items closer to faster memory devices
- Definition of the important data items
  - Data item that will be accessed frequently in near future
  - Efficient memory management == Access-aware memory management

Examples of Access-aware Memory Management Optimizations

- Access-aware Transparent Huge Pages (THP) collapse/split
  - THP reduces TLB misses (performance increase), but
    increases huge page-internal fragmentation (memory footprint bloat)
  - Idea: Use hugepages for only hot data
  - The research work was accepted to a top-tier conf (OSDI)
- Proactive reclamation
  - Memory pressure-reactive memory reclamation (Linux kernel’s default behavior) incurs unnecessarily big memory footprints and latency spikes
  - Idea: Proactively find and reclaim cold pages
  - The works from Google and Meta were accepted to a top-tier conf (ASPLOS) twice, and being used on their fleets

Access-aware Memory Management: Challenges and a Solution

- Challenges
  - Ways to know data item that will be frequently accessed in near future
    - Monitoring based prediction is reasonable, but overhead can be significant
  - Implementing each optimization from the scratch is difficult, danger, and repetitive
- An available solution: DAMO
  - A Python-written user-space tool

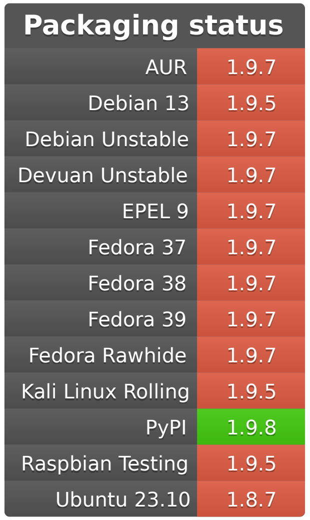
DAMO:

User-Space Tool/Python Library
for Access-Aware Profiling and Optimization

(a.k.a Data Access Monitor Operator)

DAMO: Overview

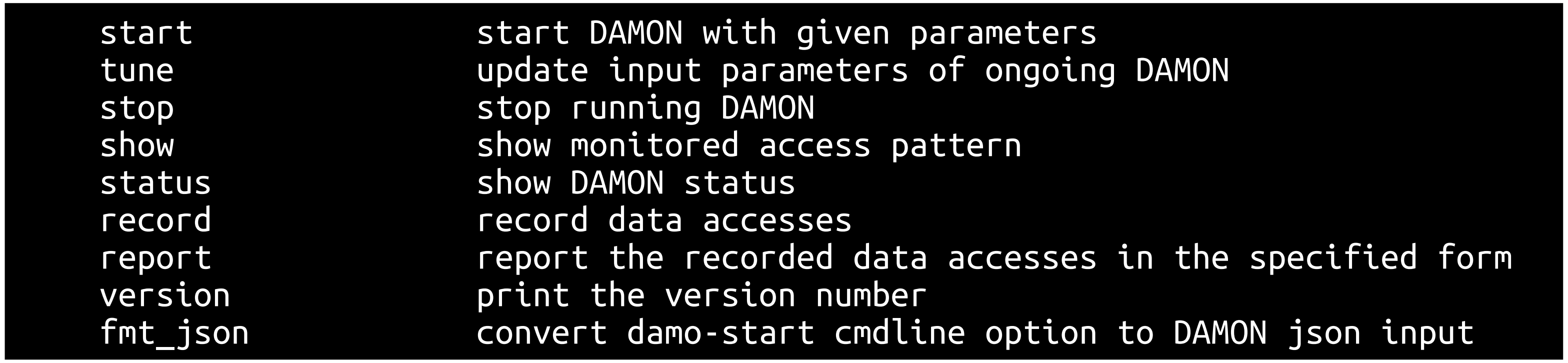
- User-space tool for Data Access MONitoring and the monitoring-based system operations
- Support all kernels having a required kernel feature
  - >=v5.15 upstream kernels
  - Distro kernels that backported the feature
    - Amazon Linux (AL) >=v5.4 kernels
    - The latest version of the feature is backported in AL v5.10 kernel
    - Android common v5.10 kernel
    - There might be unknown distro kernels that backported the feature
- Available at PyPI and a few Linux distros' package systems
  - Check the availability at repolog
  - Packaging for Linux distros are done by voluntary contributors
    - Huge appreciate to package maintainers (**michel**@ and **kokakiwi**@)!



Packaging status	
AUR	1.9.7
Debian 13	1.9.5
Debian Unstable	1.9.7
Devuan Unstable	1.9.7
EPEL 9	1.9.7
Fedora 37	1.9.7
Fedora 38	1.9.7
Fedora 39	1.9.7
Fedora Rawhide	1.9.7
Kali Linux Rolling	1.9.5
PyPI	1.9.8
Raspbian Testing	1.9.5
Ubuntu 23.10	1.8.7

DAMO’s Basic Usage: Sub-commands

- DAMO provides sub-commands for four categorized purposes
  - DAMON control: ‘`start`’, ‘`tune`’, and ‘`stop`’
  - Snapshot of monitoring results and DAMON status: ‘`show`’ and ‘`status`’
  - Record-based monitoring results profiling: ‘`record`’ and ‘`report`’
  - For convenient use and debugging of DAMO itself: ‘`version`’, ‘`fmt_json`’, and more
- For full list of sub-commands: ‘`damo --help`’



```
    start                   start DAMON with given parameters
    tune                    update input parameters of ongoing DAMON
    stop                    stop running DAMON
    show                    show monitored access pattern
    status                  show DAMON status
    record                  record data accesses
    report                  report the recorded data accesses in the specified form
    version                 print the version number
    fmt_json                convert damo-start cmdline option to DAMON json input
```

DAMO Live Demo: Recording and Visualization-based Offline Profiling

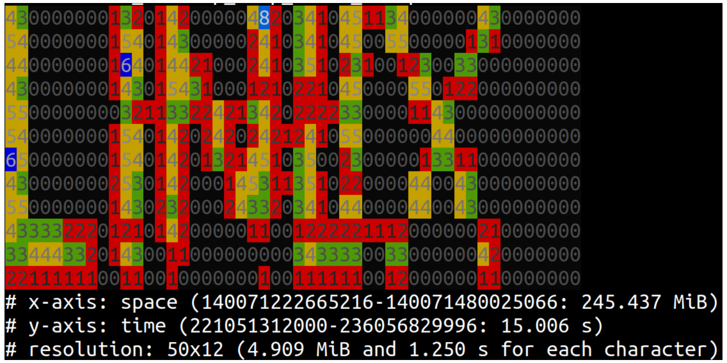
- Helps easily understanding access pattern of specific system and workloads
  - Even developers could fail at expecting their program's access pattern
    - Due to complexity, design (e.g., depends on random user request and multi-threaded), or bugs
- Collection of realistic workloads' access pattern is available

```
# git clone https://github.com/sjp38/masim -b osseu2023_demo
# cd masim && make
# damo record "./masim ./osseu2023_demo/complicated_access_pattern"
# damo report heats --heatmap stdout --stdout_heatmap_color flame --resol 12 50 \
                    --stdout_heatmap_skip_color_example
```

DAMO Live Demo: Recording and Visualization-based Offline Profiling

- Helps easily understanding access pattern of specific system and workloads
  - Even developers could fail at expecting their program’s access pattern
    - Due to complexity, design (e.g., depends on random user request and multi-threaded), or bugs
- Collection of realistic workloads’ access pattern is available

```
# git clone https://github.com/sjp38/masim -b osseu2023_demo
# cd masim && make
# damo record "./masim ./osseu2023_demo/complicated_access_pattern"
# damo report heats --heatmap stdout --stdout_heatmap_color flame --resol 12 50 \
                    --stdout_heatmap_skip_color_example
```

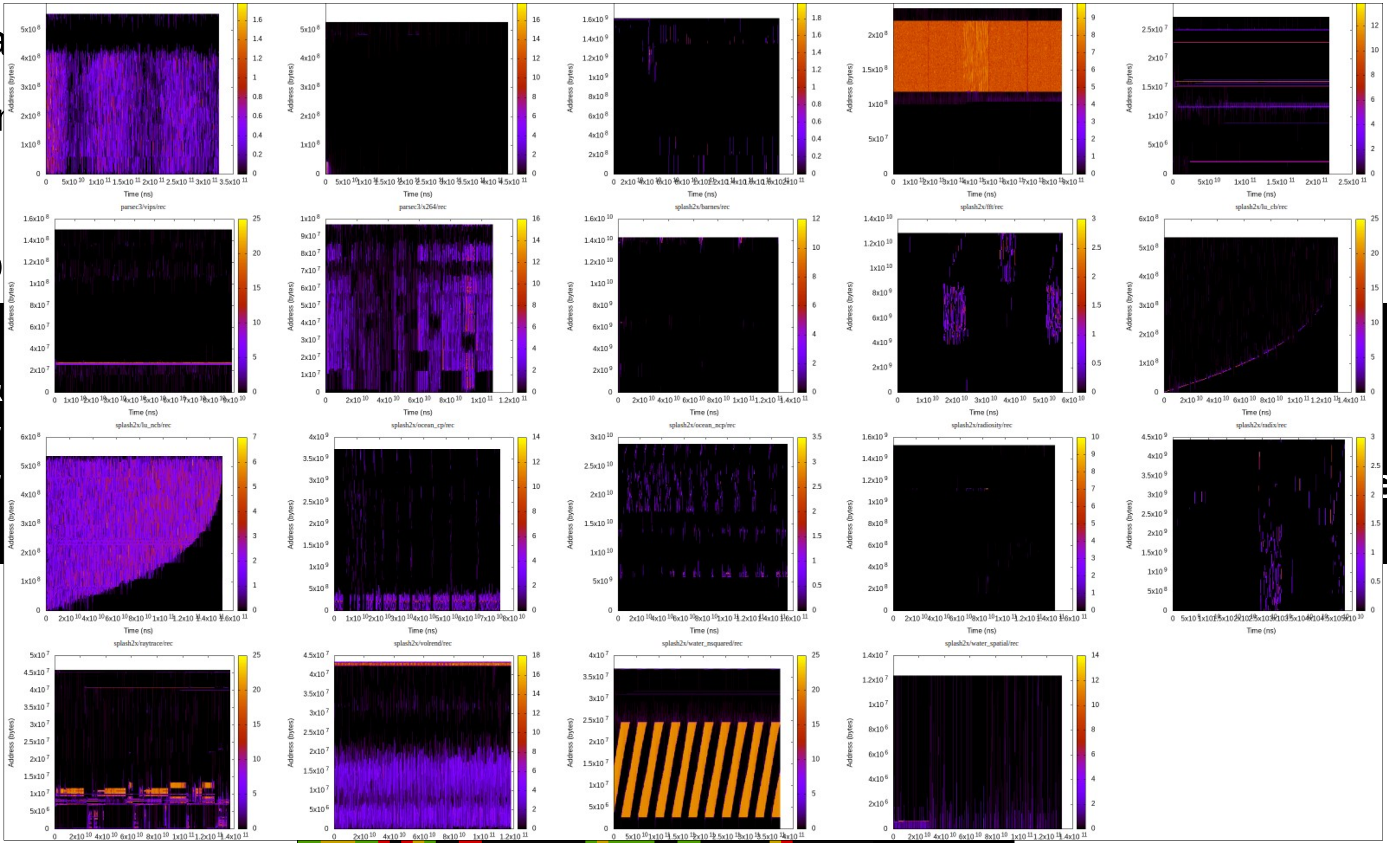


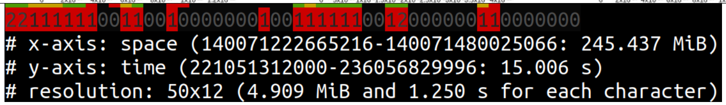
```
43000000013201420000048203410451134000000430000000
54000000015401430000024103410450055000001310000000
44000000016401442100024103510231001230033000000000
43000000014301543100012102210450000550122000000000
55000000003211332242134202222330000114300000000000
54000000015401420242024212410550000004400000000000
65000000015401420132145103500230000013311000000000
43000000025301420001453113510220000440043000000000
55000000014302320002433203410440000440043000000000
43333222012101420000011001222221112000000210000000
33444332014300110000000003433330033000000420000000
22111111001100100000001001111110012000000110000000
# x-axis: space (140071222665216-140071480025066: 245.437 MiB)
# y-axis: time (221051312000-236056829996: 15.006 s)
# resolution: 50x12 (4.909 MiB and 1.250 s for each character)
```

DAMO Live Demo: Recording and Visualization-based Offline Profiling

- Helps ea
  - Ever
    - or bugs
- Collectio

```
# git clone
# cd masim &
# damo recor
# damo repor                                              esol 12 50 \
```

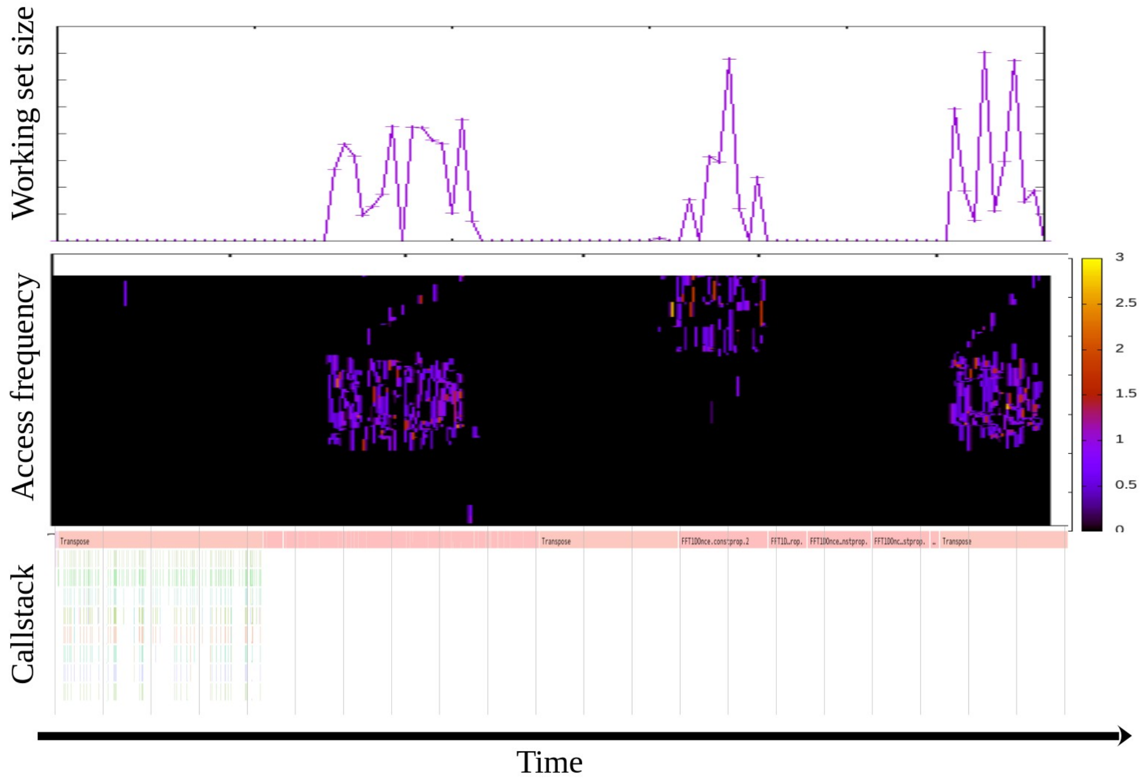




```
22111111001100100000001001111110012000000011000000
# x-axis: space (140071222665216-140071480025066: 245.437 MiB)
# y-axis: time (221051312000-236056829996: 15.006 s)
# resolution: 50x12 (4.909 MiB and 1.250 s for each character)
```

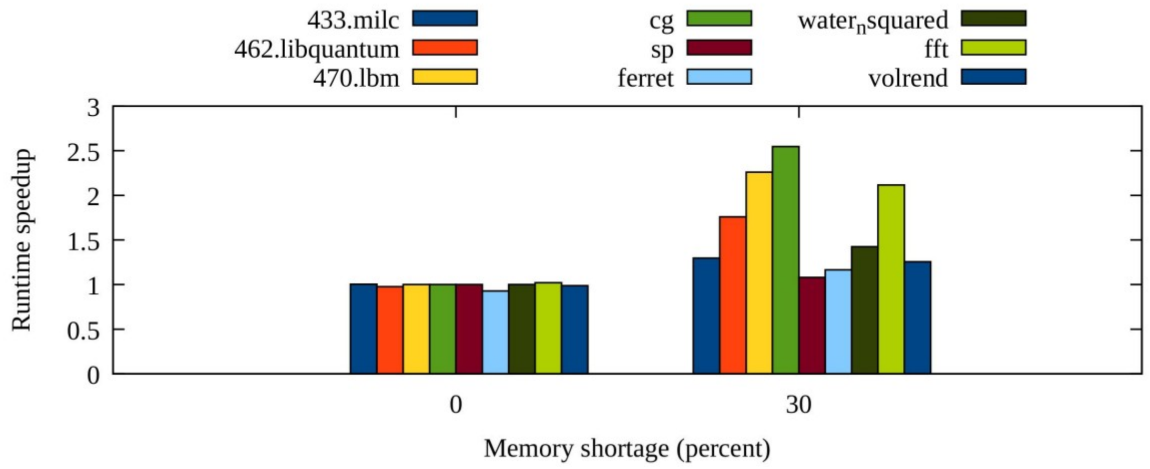
DAMO-based Offline Profiling Example

- Record and show callstacks (using perf-like tool) and access patterns (using DAMO) together
- https://sjp38.github.io/post/damon_profile_callstack_example/



DAMO-based Offline-Profiling-guided Optimization Example

- Find hot objects of each workload using DAMO-based offline-profiling and ‘mlock()’
- Up to 2.55x speedup under memory pressure
  - Shared on ksummit 2019 DAMON talk
  - Published via the proceeding of Middleware 2019 Industrial track



Retrieved from ksummit 2019 DAMON talk

DAMO Live Demo: Snapshot of Access Pattern and DAMON Status

- Useful for online profiling of access patterns

```
# damo start "./masim ./configs/stairs.cfg --repeat 5 --quiet"
# damo status
# damo show
# damo show --sort_regions_by age access_rate
# damo show --sort_regions_by age access_rate --region_box \
            --region_box_colorset flame
# damo show --access_rate 5% max
# damo stop
# killall masim
```

DAMO Live Demo: Data Access-aware Optimization

```
# damo start "./masim ./configs/stairs.cfg --repeat 5 –quiet"
# top | grep -E "RES|masim"
# #
# # Proactively reclaim memory regions that not accessed for >= 3 seconds
#
# damo tune --target_pid $(pidof masim) \
            --damos_action pageout --damos_access_rate 0% 0% --damos_age 3s max
# top | grep -E "RES|masim"
# damo stop
# killall masim
```

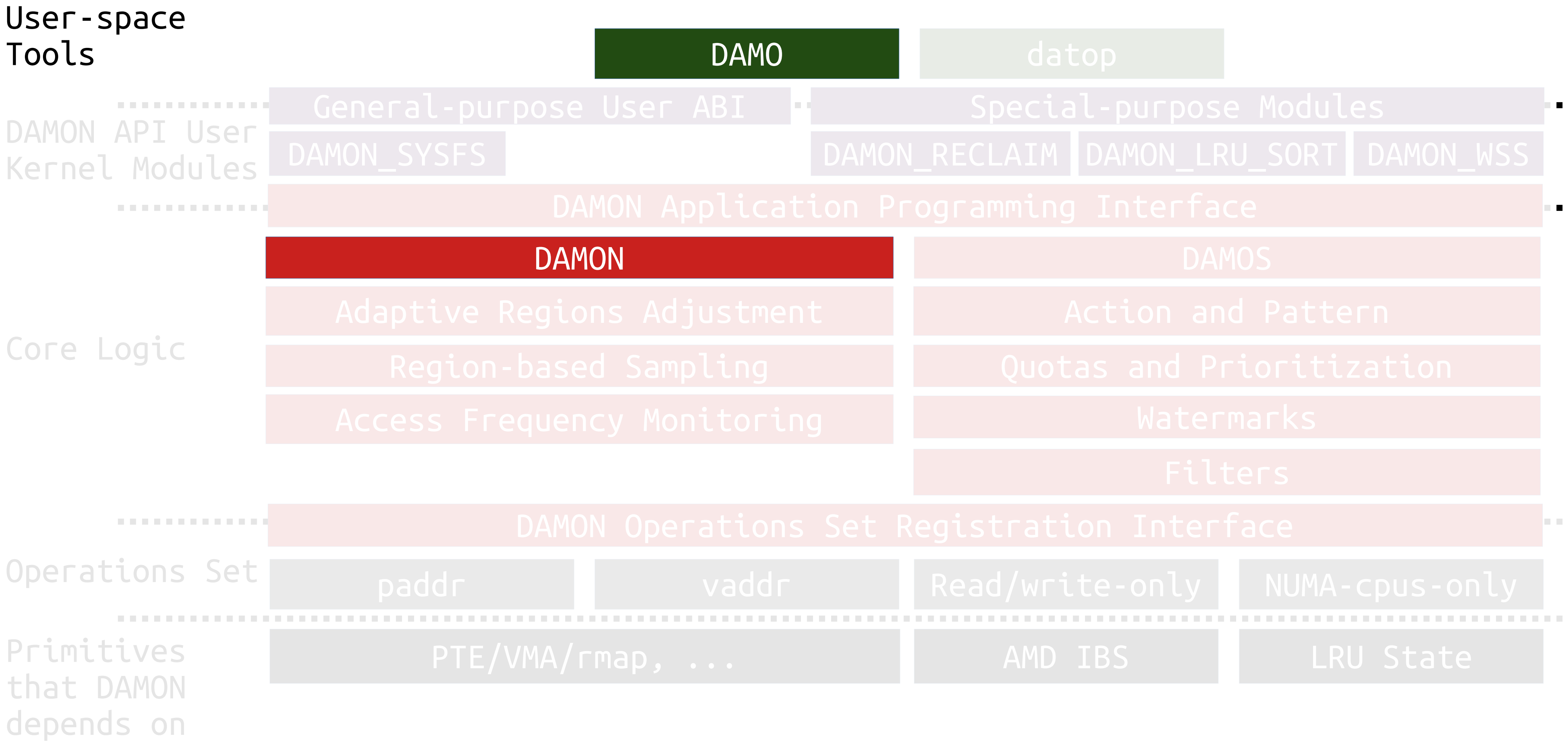
DAMON-based Access-aware System Optimization Examples

- Proactive reclamation on large serverless production environment
  - reduces up to 90% memory overhead with up to 2% CPU usage and negligible performance overhead
  - Published to a proceeding of an academic conf (HPDC’22)
- Data Access-aware THP collapse/split on realistic benchmark workloads
  - reduces up to 76% of THP overhead while keep 51% performance gain
  - Detailed results are available

DAMO: Implementation

- Delegate low level works to Linux kernel’s Data Access MONitor subsystem (DAMON)
  - Data access monitoring
  - Data access monitoring-based system operations
  - The required kernel feature is DAMON
- Implement only DAMON communication, user interface, and visualization
- Reference DAMON user-space tool implementation
  - Maintained together with DAMON
  - Support all DAMON-available Linux kernels
    - DAMON was merged in mainline since v5.15
    - DAMON was backported to AL kernel >=v5.4, and latest DAMON is ported on AL v5.10 kernel
    - DAMON was backported to Android core kernel v5.10

DAMON Stack


User-space Tools
DAMO
datop
DAMON API User Kernel Modules
General-purpose User ABI
Special-purpose Modules
DAMON_SYSFS
DAMON_RECLAIM
DAMON_LRU_SORT
DAMON_WSS
DAMON Application Programming Interface
Core Logic
DAMON
DAMOS
Adaptive Regions Adjustment
Action and Pattern
Region-based Sampling
Quotas and Prioritization
Access Frequency Monitoring
Watermarks
Filters
DAMON Operations Set Registration Interface
Operations Set
paddr
vaddr
Read/write-only
NUMA-cpus-only
Primitives that DAMON depends on
PTE/VMA/rmap, ...
AMD IBS
LRU State

DAMO-DAMON Communication

DAMO User Interface for DAMON Control

- ‘`start`’ and ‘`tune`’ receive DAMON parameters from users and pass those to DAMON
  - The commands share same command line options for DAMON parameters setup
- Partial DAMON parameters setup command line options
  - Can specify only subset of DAMON parameters
  - Provide default parameter values for unspecified parameters
  - For beginners; Convenient but restrictive (1 kdamond, 1 DAMON context, 1 monitoring target)
- Full DAMON parameters setup command line options
  - Receives full DAMON parameters via a json format input
  - For experts; Complicated but flexible

DAMON ABI

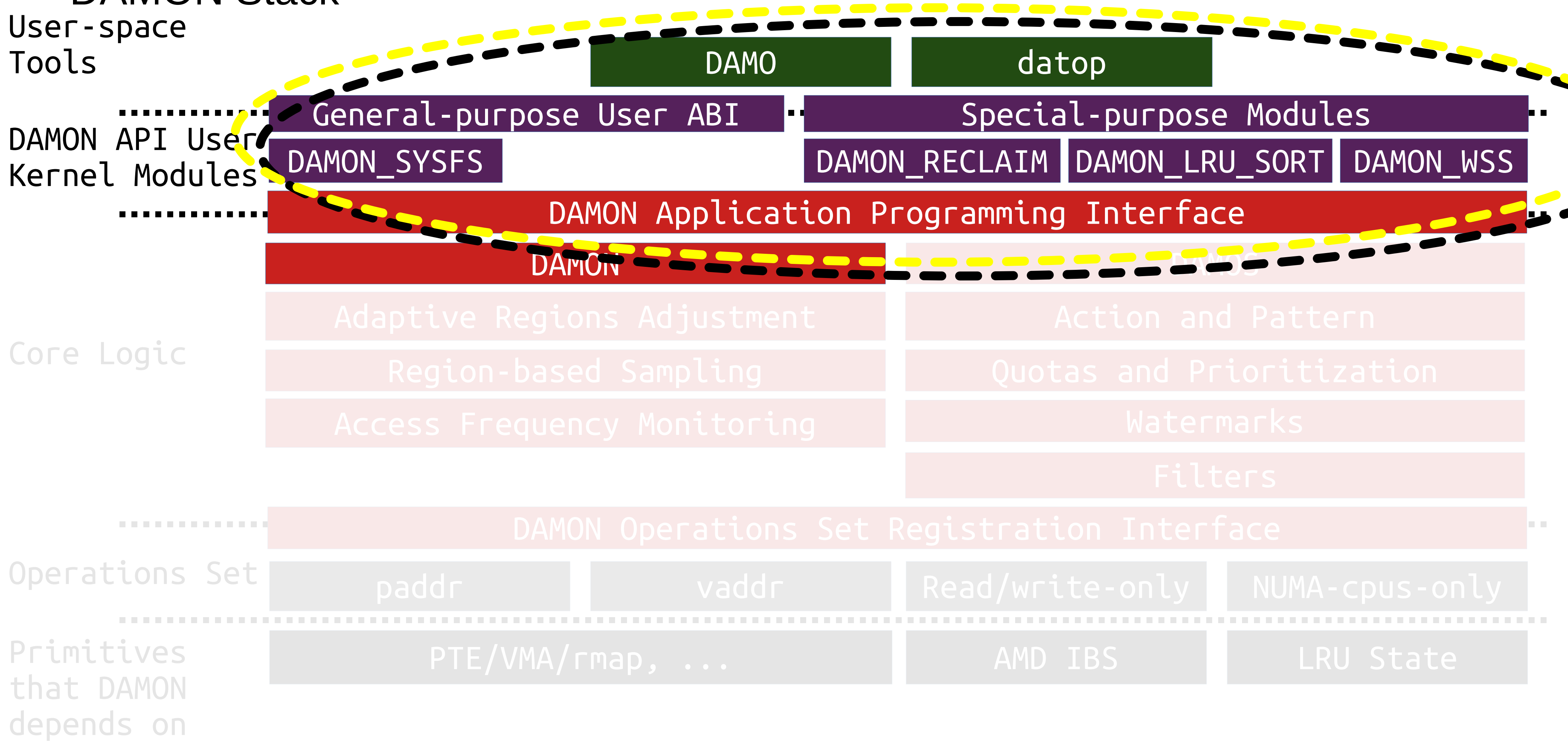
- DAMON core is a framework; Provides API to other kernel modules/subsystems
- There is a kernel module named DAMON sysfs interface (`mm/damon/sysfs.c`)
  - Implement a general-purpose DAMON ABI on `sysfs (/sys/kernel/mm/damon/)`
  - Simple enough to be used with shell commands; manual usage is discouraged though
  - DAMO passes DAMON parameters via this ABI
- DAMON kernel modules implementing special-purpose simpler ABI also available
  - `DAMON_RECLAIM` and `DAMON_LRU_SORT`

```
# 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
```

Bring Your Own DAMON User-space Tool

- DAMO is just a reference DAMON user-space tool implementation
  - Strives to be useful in many general cases and provide full features of DAMON
  - Not necessarily the one that fits all needs
  - Neither optimal; Written as a naive Python program
- Anyone can write their own DAMON user-space tools if needed
  - using DAMON ABI: like Alibaba developed DATOP (available at Github)
    - Makes sense for faster implementation (e.g., C, Go, or Rust-based tools)
  - using DAMO’s core modules (wraps DAMON ABI) as a library
    - DAMO’s many features are implemented in this way
      (Refer to damo_start.py, damo_stop.py and damo_record.py for example usage)
    - Makes sense for more specialized features
    - The interface is unstable; merging your tool into DAMO can be an option (like upstreamed drivers)

DAMON Stack


User-space Tools
DAMO
datop
DAMON API User Kernel Modules
General-purpose User ABI
Special-purpose Modules
DAMON_SYSFS
DAMON_RECLAIM
DAMON_LRU_SORT
DAMON_WSS
DAMON Application Programming Interface
DAMON
DAMOS
Core Logic
Adaptive Regions Adjustment
Action and Pattern
Region-based Sampling
Quotas and Prioritization
Access Frequency Monitoring
Watermarks
Filters
DAMON Operations Set Registration Interface
Operations Set
paddr
vaddr
Read/write-only
NUMA-cpus-only
Primitives that DAMON depends on
PTE/VMA/rmap, ...
AMD IBS
LRU State

DAMON Parameters for Access Monitoring (DAMON)

Getting Full Parameter (json) Input

- Writing json input from the scratch is boring; helper commands are available
- ‘`damo fmt_json`’
  - Convert partial DAMON parameters setup input to the full parameters input
- ‘`damo status --json`’
  - Convert parameters of running DAMON to the full parameters input
- Note: Above commands expose not only parameters but also status of DAMON
  - fields for the status can be omitted for DAMON full parameters setup input

Kdamonds and DAMON Contexts

- Kdamond (item of `[“kdamonds”]` of the json input)
  - DAMON worker thread
- DAMON Context (item of `[“contexts”]` of each kdamond in the json input)
  - Data structure for the monitoring requests and running status
- Each kdamond should have at least one DAMON context
  - Only single context per kdamond is supported for now
- Can scale-out with multiple kdamonds
  - For more CPU resource or running different monitoring requests

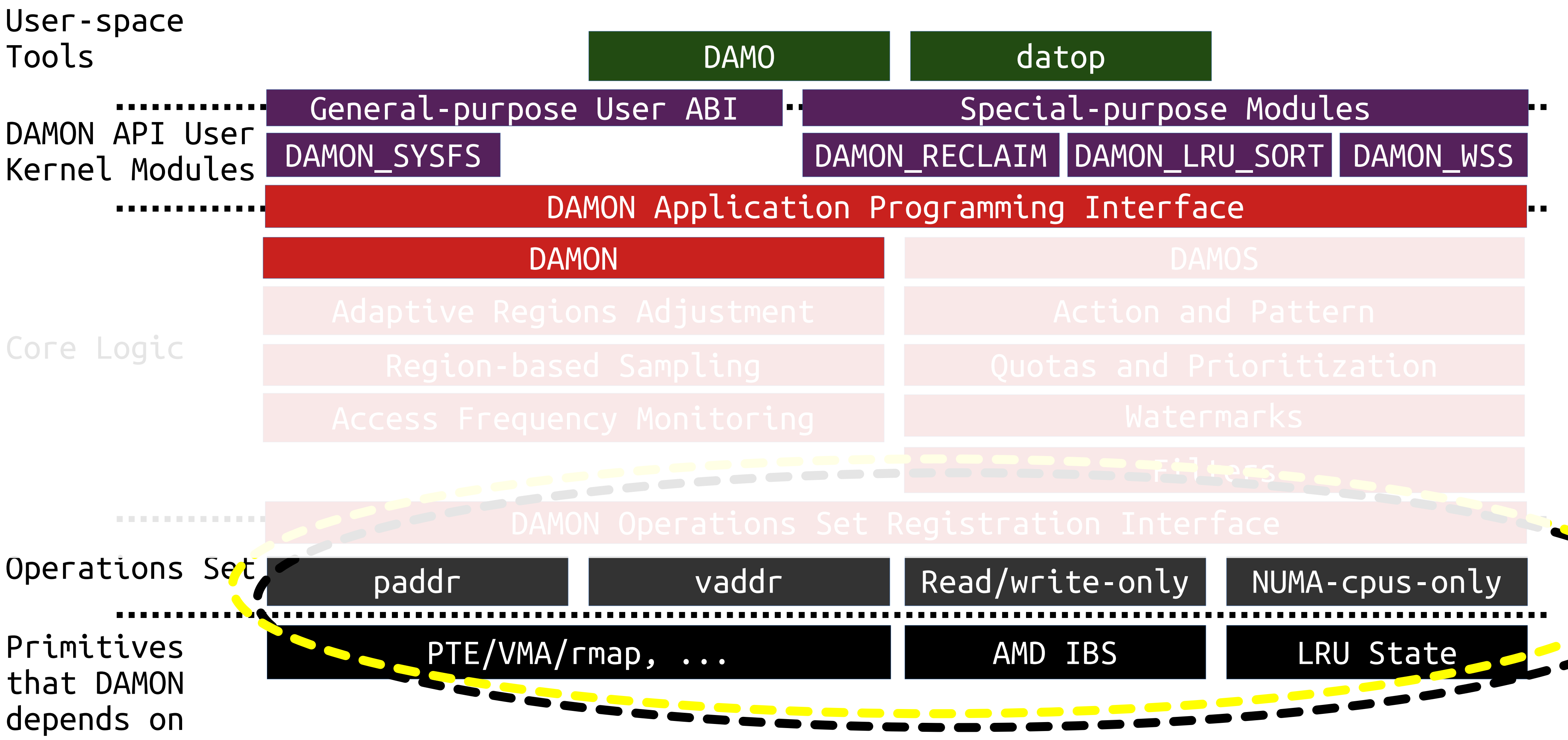
```
{"kdamonds":
    [
        {"contexts": [{[…]}]}
    ]
}
```

Operations Set

- [“ops”] of DAMON context in json input
- Data monitoring primitive level operation implementations
- Users can select for given use case
  - ‘`paddr`’: for monitoring the physical address space using page table accessed bits
  - ‘`vaddr`’: for monitoring virtual address spaces using page table accessed bits
- Selections and extensions are available via “*operations set registration interface*”
- More operations set can be added in future
    - e.g., for moitoring the physical address space using AMD IBS

```
"contexts": [ {"ops": "paddr",
```

DAMON Stack


User-space Tools
DAMO
datop
DAMON API User Kernel Modules
General-purpose User ABI
Special-purpose Modules
DAMON_SYSFS
DAMON_RECLAIM
DAMON_LRU_SORT
DAMON_WSS
DAMON Application Programming Interface
Core Logic
DAMON
DAMOS
Adaptive Regions Adjustment
Action and Pattern
Region-based Sampling
Quotas and Prioritization
Access Frequency Monitoring
Watermarks
Filters
DAMON Operations Set Registration Interface
Operations Set
paddr
vaddr
Read/write-only
NUMA-cpus-only
Primitives that DAMON depends on
PTE/VMA/rmap, ...
AMD IBS
LRU State

Monitoring Targets

- items of `[“targets”]` of DAMON context in json input
- The target address spaces and address ranges of those to monitor the accesses to it
  - ‘`pid`’ is only for ‘`vaddr`’ ops (virtual address space of the pid is the monitoring target)
  - ‘regions’ specifies address ranges of each address space to monitor
    - In case of ‘`vaddr`’ ops, it periodically resets the target address ranges to cover all mapped regions

```
"contexts": [
    {
        "targets": [
            {
                "pid": null,
                "regions": [
                   {"start": "4,294,967,296", "end": "136,299,151,359"}
                ]
             }
         ],
```

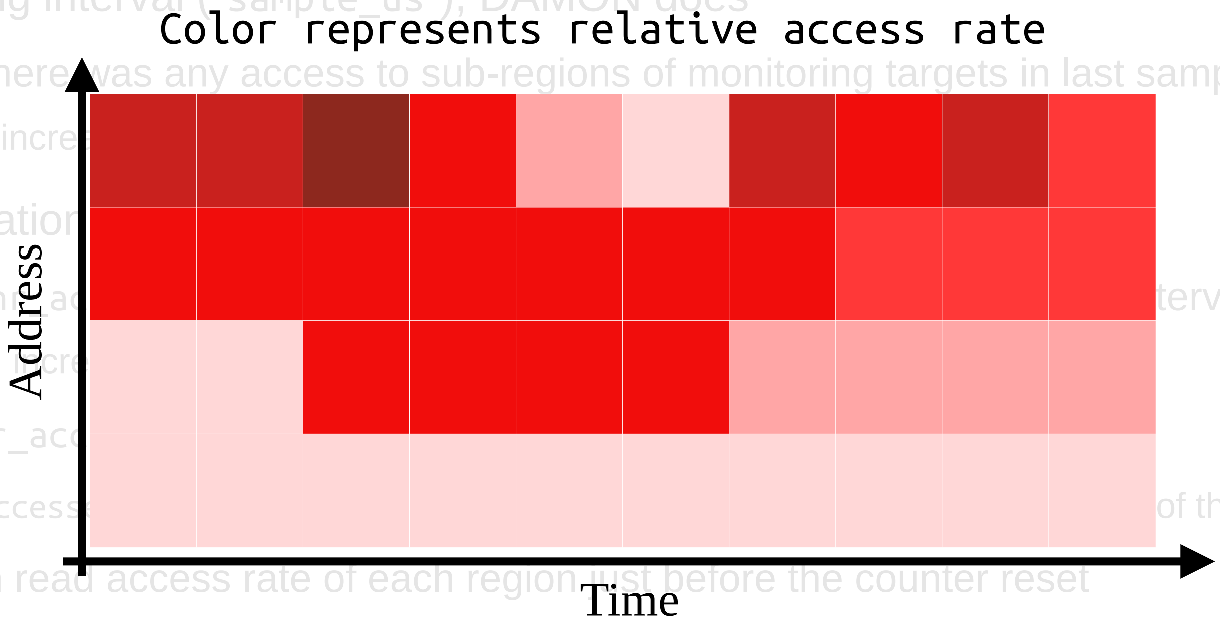
Monitoring Intervals (1/3)

- `[“intervals”]` of DAMON context in json input
- Every sampling interval (“`sample_us`”), DAMON does
  - Check if there was any access to sub-regions of monitoring targets in last sampling interval
    - If so, increase a counter (‘`nr_accesses`’) of the sub-region
- Every aggregation interval (“`aggr_us`”), DAMON does
  - Check if ‘`nr_accesses`’ significantly changed from that of last aggregation interval
    - If not, increase a counter (‘`age`’) of the region, reset the counter otherwise
  - Resets ‘`nr_accesses`’
    - ‘`nr_accesses`’ / (aggregation interval / sampling interval) is called “*access rate*” of the region
  - Users can read access rate of each region just before the counter reset
- This is called “*Access Frequency Monitoring*”

```
"contexts": [{"intervals": {"sample_us": "5 ms", "aggr_us": "100 ms", "ops_update_us": "1 s"},
```

Monitoring Intervals (1/3)

- [“`intervals`”] of DAMON context in json input
- Every sampling interval (“`sample_us`”), DAMON does
  - Check if there was any access to sub-regions of monitoring targets in last sampling interval
    - If so, incre
- Every aggregation
  - Check if ‘`nr_ac` terval
    - If not, incre
  - Resets ‘`nr_acc`
    - ‘`nr_access` of the region
  - Users can read access rate of each region just before the counter reset
- This is called “*Access Frequency Monitoring*”



```
"contexts": [{"intervals": {"sample_us": "5 ms", "aggr_us": "100 ms", "ops_update_us": "1 s"},
```

Monitoring Intervals (2/3)

- Sub-region are defined as address range of pages having similar access rate
  - DAMON checks access to only one randomly-picked sample page in each region
  - Monitoring overhead gets depend on access pattern (number of regions), not size of memory
  - This is called “*Regions-based Access Sampling*”
- Monitoring accuracy depends on if regions are well identified as defined
  - Randomly split regions to make those aligned with the definition
    - Unnecessary split increases monitoring overhead
  - Merge adjacent regions of similar nr_accesses for every aggregation interval
    - Revert unnecessary splits
- This is called “*Adaptive Regions Adjustment*”

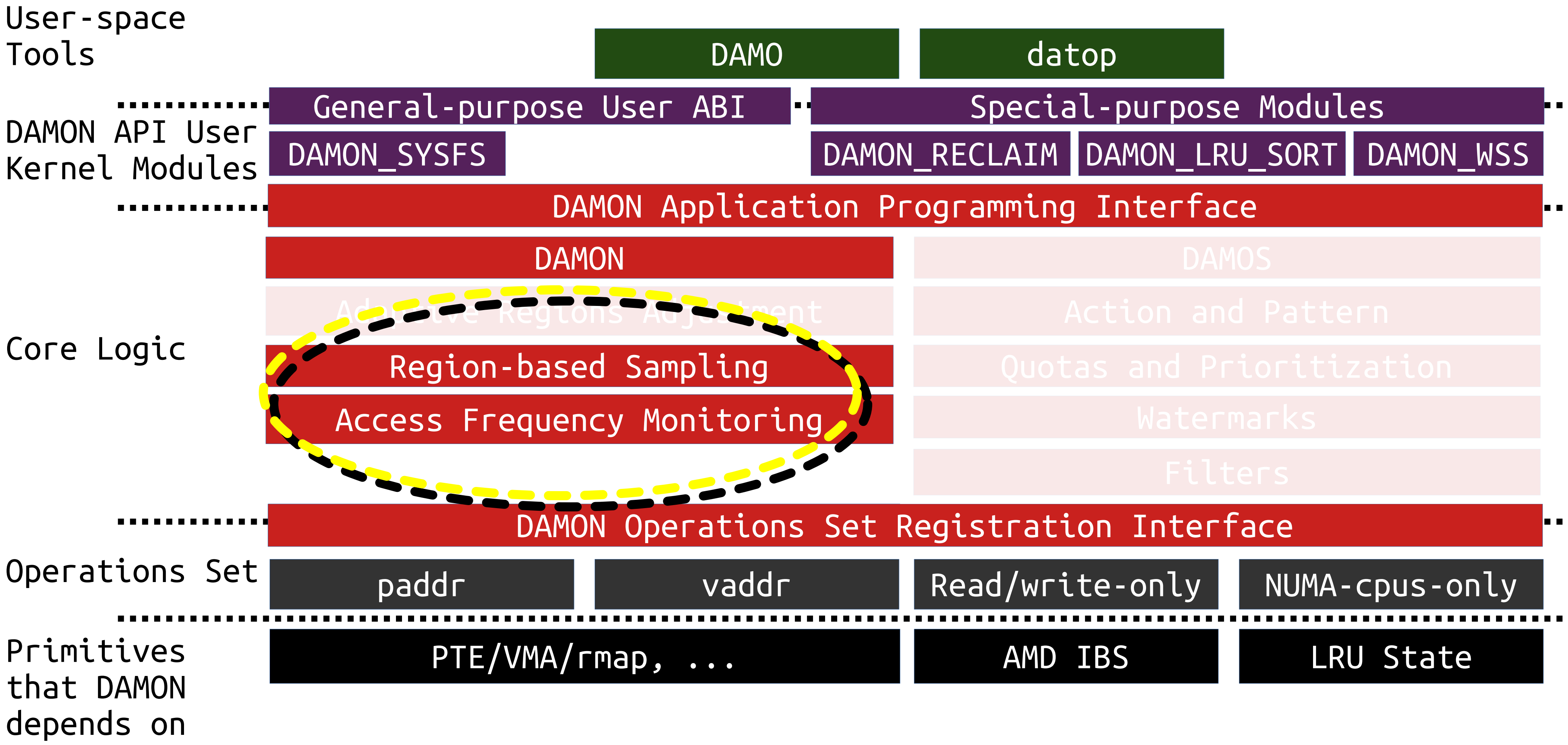
```
"contexts": [{"intervals": {"sample_us": "5 ms", "aggr_us": "100 ms", "ops_update_us": "1 s"},
```

Monitoring Intervals (3/3)

- DAMON signals “operations set” every Ops update interval (ops_update_us) for any needed updates of themselves
  - ‘`vaddr`’ ops updates target monitoring address ranges to cover updated mapped regions
  - ‘`paddr`’ do nothing for the signal

```
"contexts": [{"intervals": {"sample_us": "5 ms", "aggr_us": "100 ms", "ops_update_us": "1 s"},
```

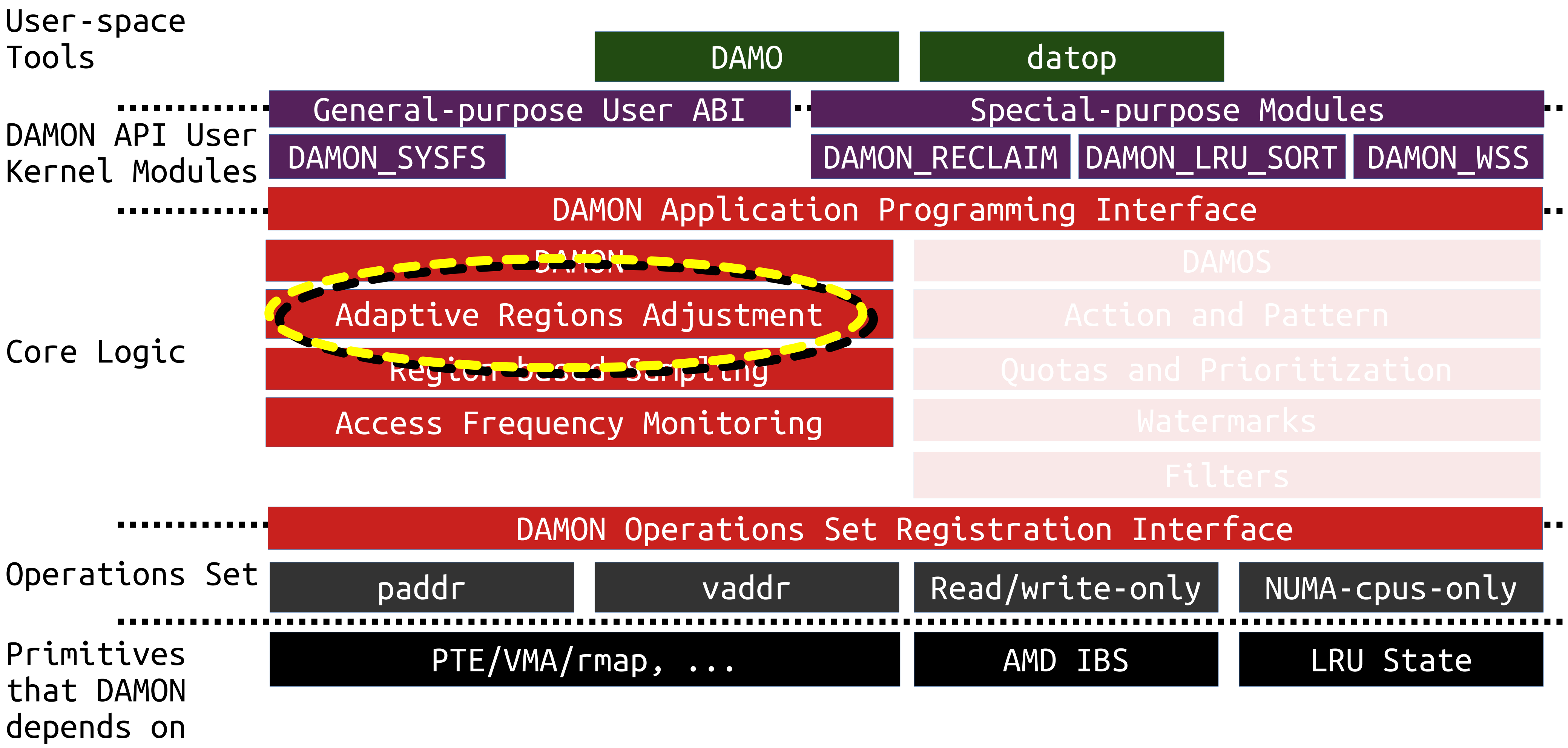
DAMON Stack


User-space Tools
DAMO
datop
DAMON API User Kernel Modules
General-purpose User ABI
Special-purpose Modules
DAMON_SYSFS
DAMON_RECLAIM
DAMON_LRU_SORT
DAMON_WSS
DAMON Application Programming Interface
Core Logic
DAMON
DAMOS
Adaptive Regions Adjustment
Action and Pattern
Region-based Sampling
Quotas and Prioritization
Access Frequency Monitoring
Watermarks
Filters
DAMON Operations Set Registration Interface
Operations Set
paddr
vaddr
Read/write-only
NUMA-cpus-only
Primitives that DAMON depends on
PTE/VMA/rmap, ...
AMD IBS
LRU State

Min/Max Number of Regions

- ‘`min`’, and ‘`max`’ field of `[‘nr_regions’]` of DAMON context in json input
- Merge and Split are avoided if it can make the number of regions out of the bound
  - Work as accuracy lower-limit (‘min’) and overhead upper-limit (‘max’) of monitoring
- This is a part of “*Adaptive Regions Adjustment*”
- Effective limits, but not intuitive knobs; Future improvements are needed

DAMON Stack


User-space Tools
DAMO
datop
DAMON API User Kernel Modules
General-purpose User ABI
Special-purpose Modules
DAMON_SYSFS
DAMON_RECLAIM
DAMON_LRU_SORT
DAMON_WSS
DAMON Application Programming Interface
Core Logic
DAMON
Adaptive Regions Adjustment
Region-based Sampling
Access Frequency Monitoring
DAMOS
Action and Pattern
Quotas and Prioritization
Watermarks
Filters
DAMON Operations Set Registration Interface
Operations Set
paddr
vaddr
Read/write-only
NUMA-cpus-only
Primitives that DAMON depends on
PTE/VMA/rmap, ...
AMD IBS
LRU State

DAMON Parameters for Access Monitoring-based Memory Management

(a.k.a DAMOS: DAMon-based Operation Schemes)

DAMOS Schemes

- Yet another DAMON core feature for no-code access-aware memory management
  - Find memory regions of specific access pattern and apply a user-requested action
- Items of [“schemes”] of DAMON context in json input
  - No scheme by default
  - Can be added with ‘--damos_*’ partial DAMON parameter setup command line options
    - e.g., ‘damo fmt_json --damos_action stat’

```
    "contexts": [
        {
            "schemes": [
```

DAMOS Action

- `[“action”]` of DAMOS scheme in json input
- The system operation that users want to apply to pages in regions of specific access pattern
- Supports actions including
  - ‘`pageout`’: Page out the pages
  - ‘`hugepage`’: Hint THP collapse/split deamon to collapse the pages to huge pages
  - ‘`nohugepage`’: Hint THP collapse/split deamon to split the pages to regular pages
  - ‘`lru_prio`’: Make the pages last eviction candidate under future memory pressure
  - ‘`lru_deprio`’: Make the pages first eviction candidate under future memory pressure
  - ‘`stat`’: Do nothing but count statistics

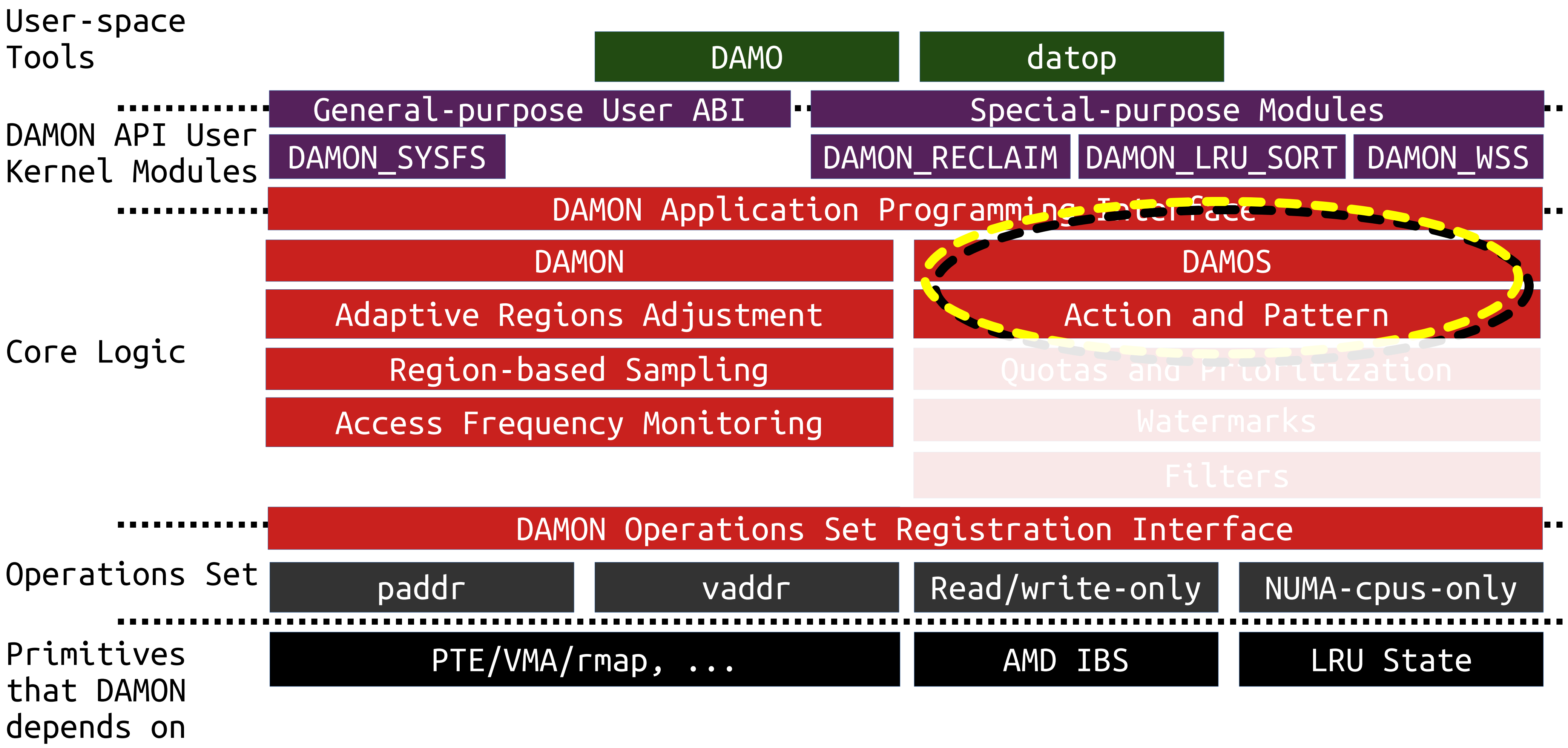
```
"contexts": [{"schemes": [{"action": "stat",
```

DAMOS Access Pattern

- [“access pattern”] of each scheme in json input
- The access pattern of the interest
- Constructed with three ranges
  - “sz_bytes”: size of the region
  - “nr_accesses”: access rate
  - “age”: time that the sz_bytes and nr_accesses have maintained

```
"schemes": [
    {
        "access_pattern": {
        "sz_bytes": {"min": "0 B", "max": "max"},
        "nr_accesses": {"min": "0 %", "max": "100 %"},
        "age": {"min": "0 ns", "max": "max"}
    },
```

DAMON Stack


User-space Tools
DAMO
datop
General-purpose User ABI
Special-purpose Modules
DAMON API User Kernel Modules
DAMON_SYSFS
DAMON_RECLAIM
DAMON_LRU_SORT
DAMON_WSS
DAMON Application Programming Interface
Core Logic
DAMON
DAMOS
Adaptive Regions Adjustment
Action and Pattern
Region-based Sampling
Quotas and Prioritization
Access Frequency Monitoring
Watermarks
Filters
DAMON Operations Set Registration Interface
Operations Set
paddr
vaddr
Read/write-only
NUMA-cpus-only
Primitives that DAMON depends on
PTE/VMA/rmap, ...
AMD IBS
LRU State

DAMOS Schemes: Quotas

- `[“quotas”]` of each scheme in json input
- Limit maximum CPU usage or amount of memory region for applying DAMOS action
  - e.g., 10 ms per 1 s and 1 GiB per 1s for applying the scheme action
- Under the quota, regions are prioritized based on access pattern
  - The action is applied to more prioritized regions first
  - Prioritization mechanism depends on the action: Colder pages are prioritized for `pageout` action
  - Access pattern elements can get different priority weights

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

DAMOS Schemes: Quotas

- `[“quotas”]` of each scheme in json input
- Limit maximum CPU usage or amount of memory region for applying DAMOS action
  - e.g., 10 ms per 1 s and 1 GiB per 1s for applying the scheme action
- Under the quota, regions are prioritized based on access pattern
  - The action is applied to more prioritized regions first
  - Prioritization mechanism depends on the action: Colder pages are prioritized for `pageout` action
  - Access pattern elements can get different priority weights

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

DAMOS Schemes: Watermarks

- `[“watermarks”]` of each scheme in json input
- [de]activate DAMOS schemes with specific system metric thresholds
  - e.g., Activate proactive reclamation if free memory rate becomes lower than 30%, Deactivate it if free memory rate becomes higher than 50%
  - Global free memory rate is supported as the system metric as of this talk

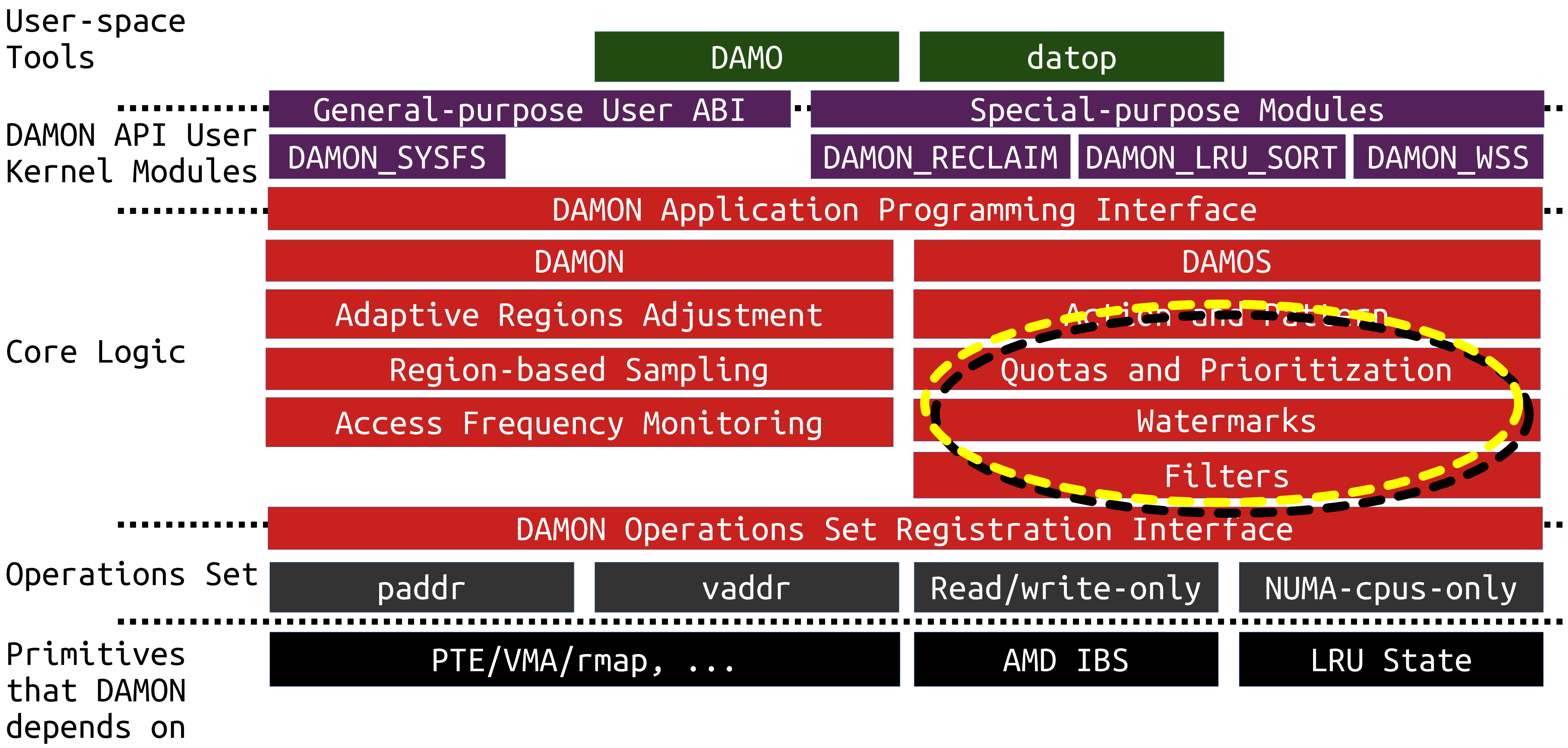
```
"watermarks": {
    "metric": "free_mem_rate",
    "interval_us": "5 s",
    "high_permil": "50 %",
    "mid_permil": "30 %",
    "low_permil": "0 %"
},
```

DAMOS Schemes: Filters

- `[“filters”]` of each scheme in json input
- Let DAMOS actions applied to only pages of specific conditions
  - Backing content (anonymous or file-backed), belonging memory cgroup, address ranges, and monitoring target are supported
  - e.g., Proactively reclaim only file-backed pages in specific NUMA node of specific cgroups

```
"filters": [
    {
        "filter_type": "anon",
        "matching": true,
        "memcg_path": null,
        "address_range": null,
        "damon_target_idx": null
    }
]
```

DAMON Stack


User-space Tools
DAMO
datop
DAMON API User Kernel Modules
General-purpose User ABI
Special-purpose Modules
DAMON_SYSFS
DAMON_RECLAIM
DAMON_LRU_SORT
DAMON_WSS
DAMON Application Programming Interface
Core Logic
DAMON
DAMOS
Adaptive Regions Adjustment
Action and Pattern
Region-based Sampling
Quotas and Prioritization
Access Frequency Monitoring
Watermarks
Filters
DAMON Operations Set Registration Interface
Operations Set
paddr
vaddr
Read/write-only
NUMA-cpus-only
Primitives that DAMON depends on
PTE/VMA/rmap, ...
AMD IBS
LRU State

DAMON Community

Community Members

- Anyone interested in DAMON is a member
- Some people from companies including Alibaba, AMD, AWS, DigitalOcean, Google, Hocus, Huaweii, IBM, Meta, Oppo, Oracle, etc seems using or experimenting it
  - Amazon Linux ported initial version of DAMON in their >=v5.4 kernels
  - Latest DAMON is continuously ported in Amazon Linux v5.10 kernel
  - Android common kernel ported and enabled DAMON_RECLAIM
  - Hocus published their research on DAMON-based proactive reclamation
- 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 rapidly evolving
  - It might not perfectly fit for your use case
- Don’t forgive it or wait for someone to implement it; make your voice
  - Report your use case and challenges
  - Ask questions and request features
  - Show your interest to known future works
  - Test and share the results
  - Send patches

Conclusion

- More data access-aware system operations are needed
- DAMO can be a solution
  - DAMO stands on Linux kernel subsystem called DAMON
- 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**

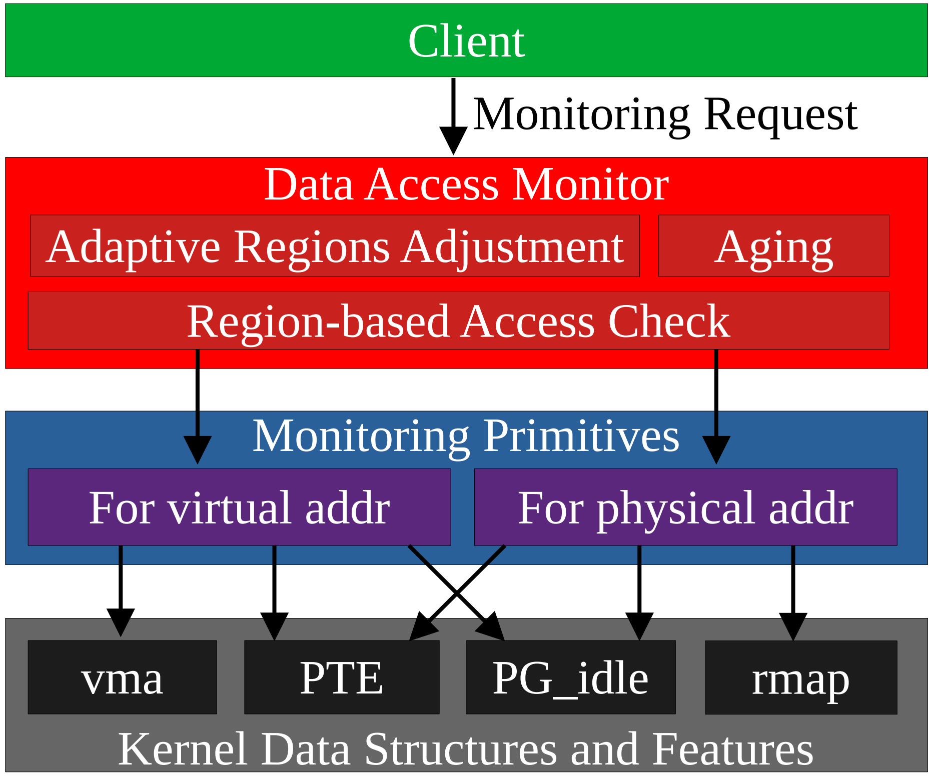
Backup Slides

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)
```

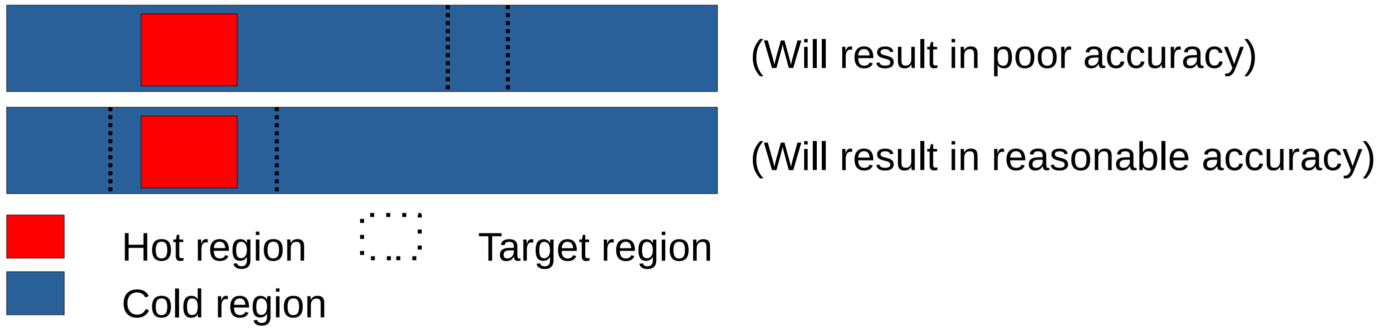
DAMON: Resulting Architecture

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



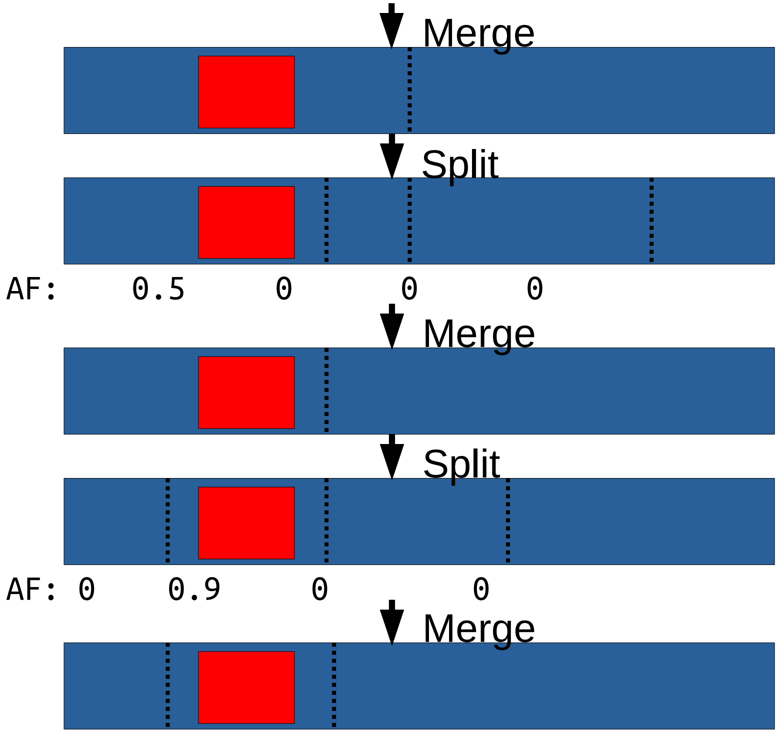
Region-based Sampling

- 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
Cold region
Target 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



AF: (Observed) Access Frequency

🟥 Hot region (AF 1.0)

🟦 Cold region (AF 0)

⬚ Target region