The International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC),
Minneapolis, Minnesota, United States, June 27 – July 1, 2022

DAOS:
Data Access-aware Operating System

SeongJae Park
amazon

Madhuparna Bhowmik
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

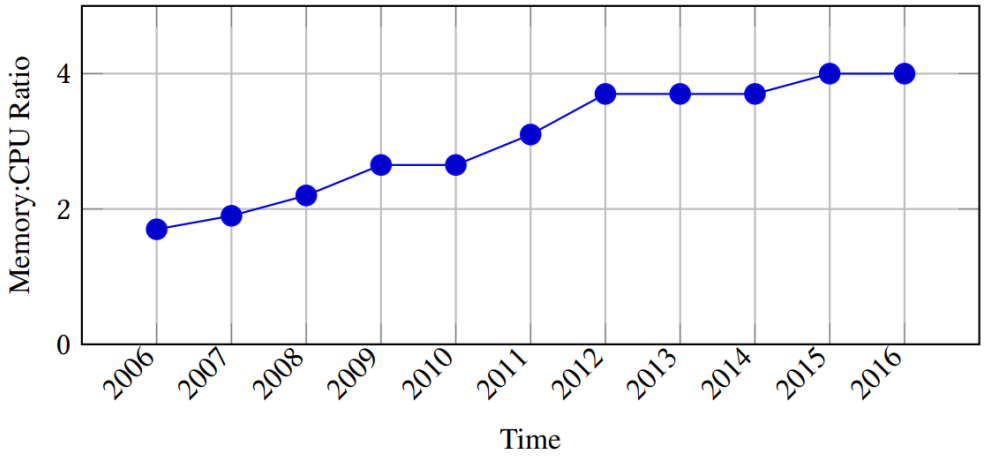
Alexandru Uta
amazon

Disclaimer

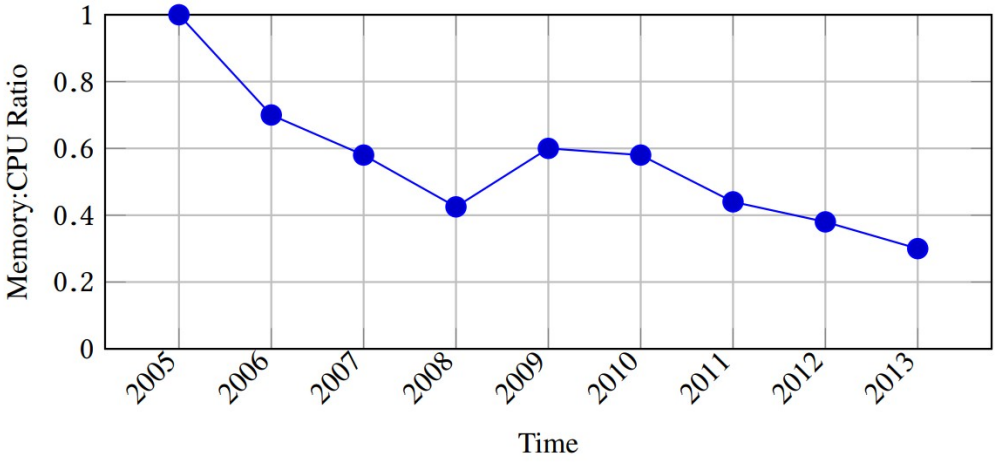
- The views expressed herein are those of the speaker; they do not reflect the views of his employers

The Trends

- Demands for memory is increasing
  - Modern workloads (e.g., cloud, big data, ML, …) use larger working set
- Size of DRAM in each machine is growing slower than that
  - Due to issues including heat/power/cost/supply
  - Tiered memory is available
- DRAM over-committed systems becoming prevalent



For AWS instances of m* types
(virtual machines: demand)



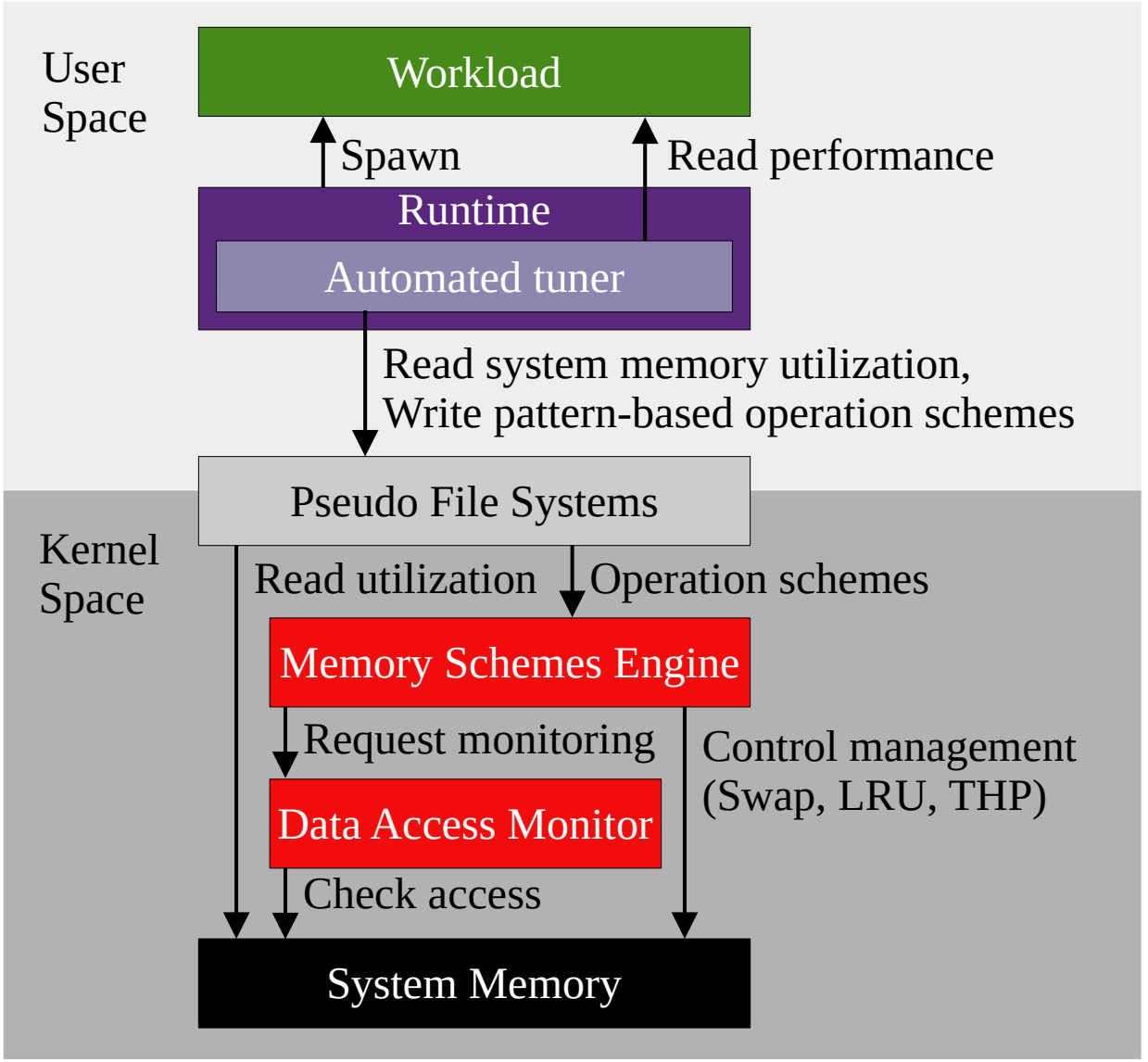
For multiple server generations
(physical machines: supply)

(Images retrieved from https://oatao.univ-toulouse.fr/24818/1/nitu_24818.pdf)

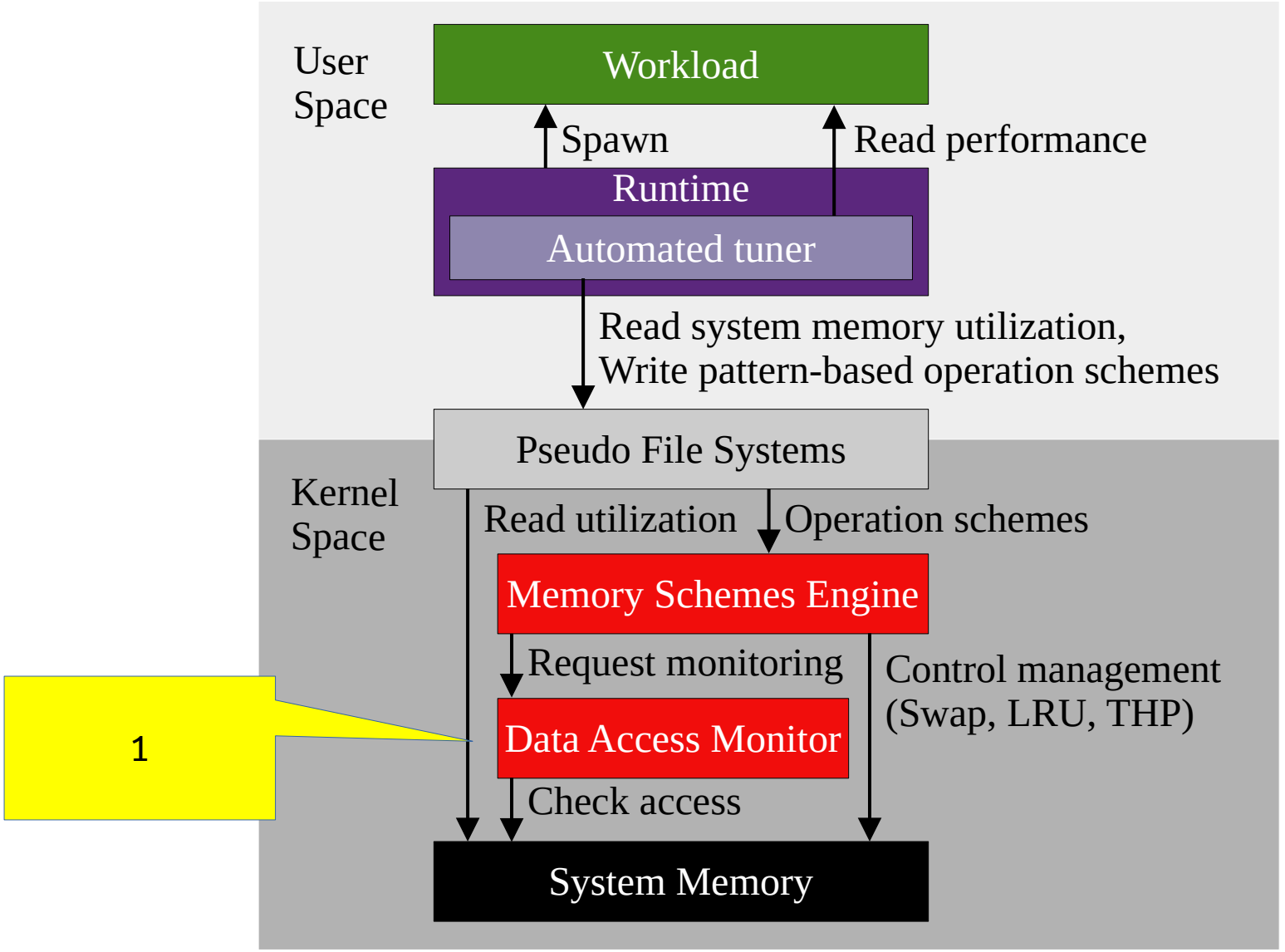
We Need Data Access-aware Operating System (DAOS)

- Because DAOS can predict future memory usage better
- Because it helps making better data management decision
- Because it can improve memory efficiency and performance
- Because DRAM is a major infrastructure expense

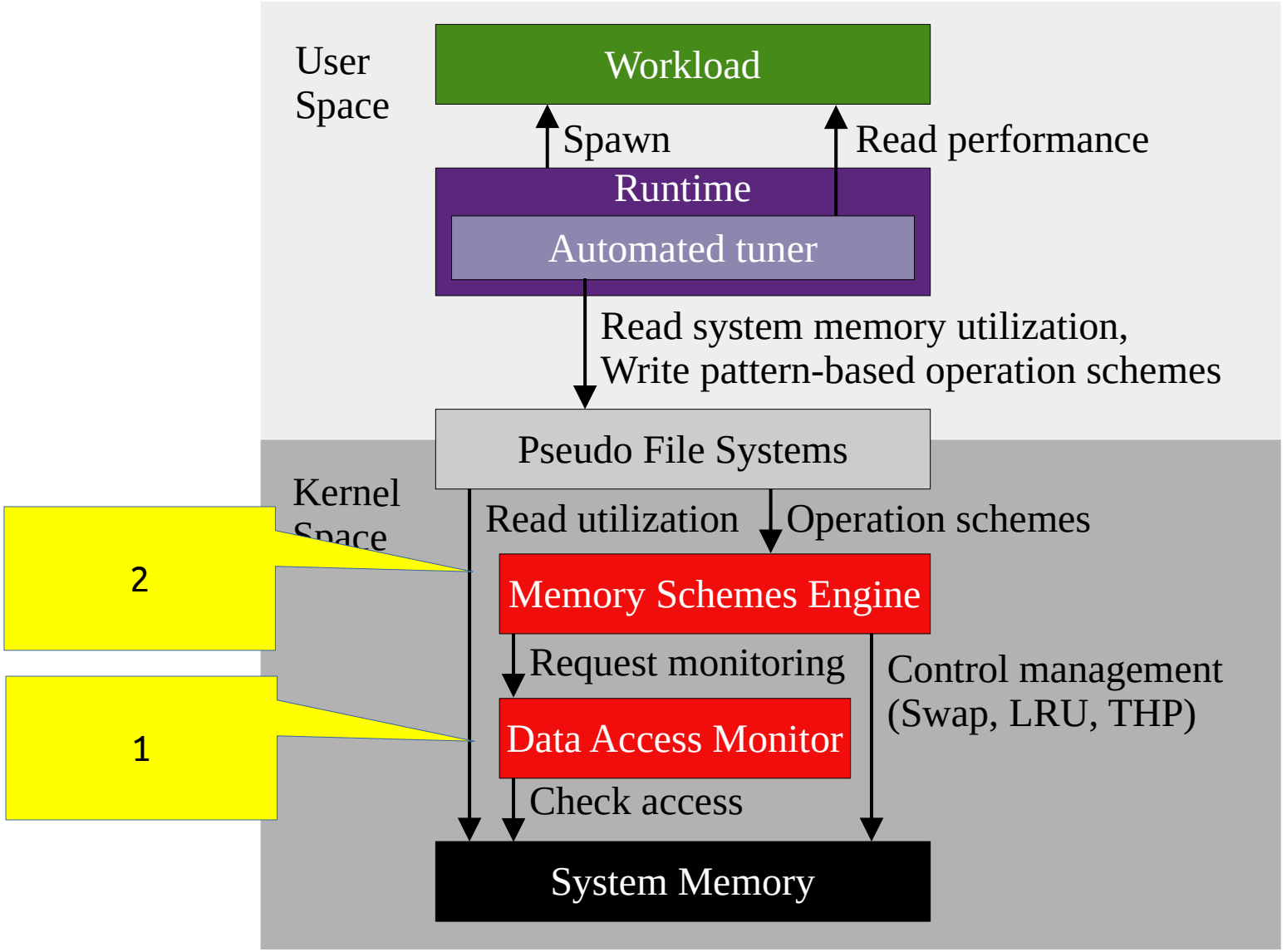
DAOS: Three Simple Pieces



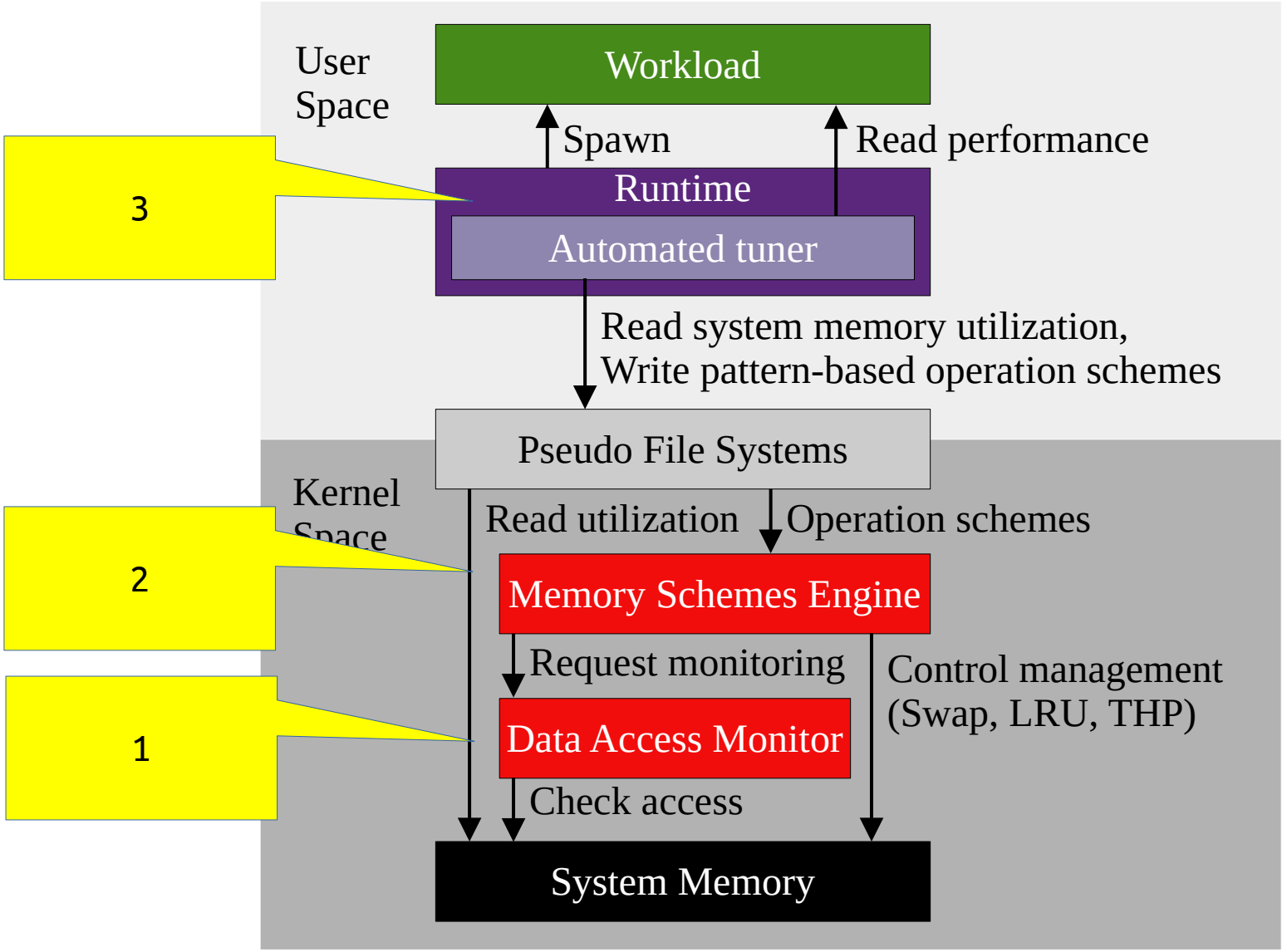
DAOS: Three Simple Pieces



DAOS: Three Simple Pieces

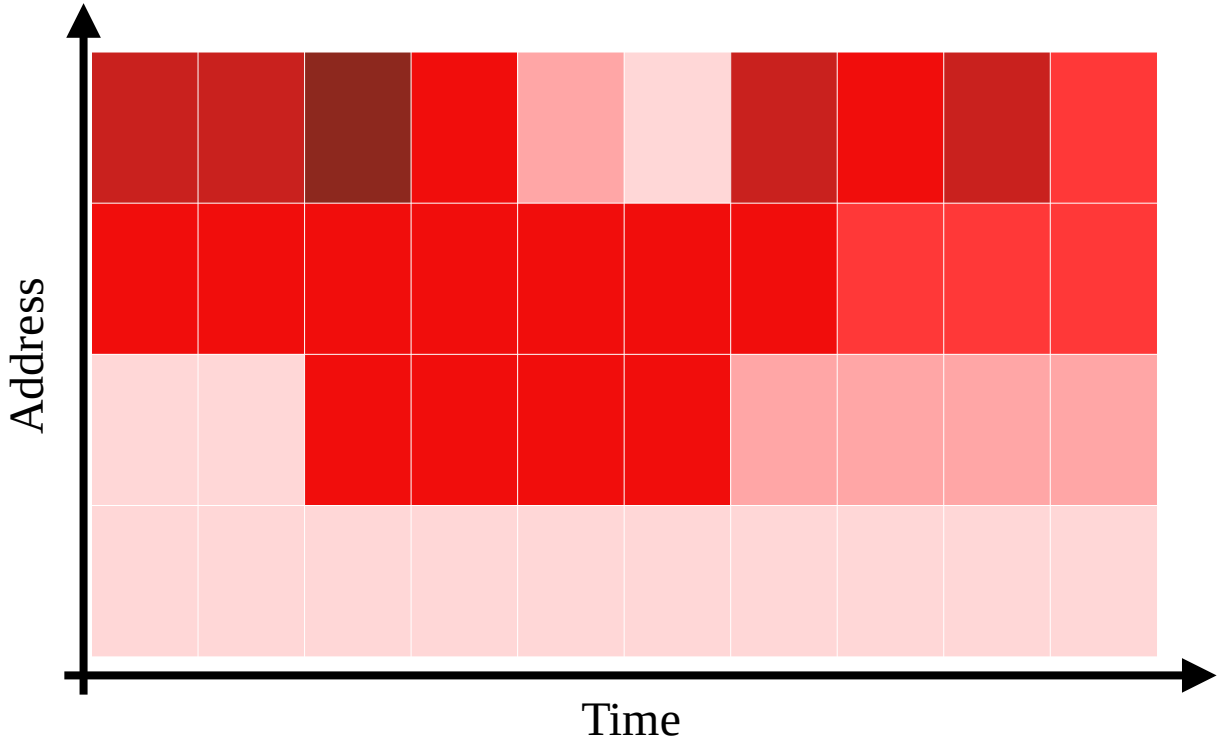


DAOS: Three Simple Pieces



DAMON: Data Access MONitor

- Data access monitoring component for DAOS
  - Provides access frequency of each address range
  - DAOS gets data access patterns of the system online using this component



Existing Data Access Monitoring Techniques

- Instrumentation-based
  - Hook/trace every memory access
  - Incur unacceptably high overhead for unnecessarily precise information
  - Some static analysis based technique reduces the overhead but recompiling is required
- Page table access bit tracking
  - Repeatedly clear and read page table entries' accessed bits
  - The overhead is not negligible and arbitrarily grows as the size of the target memory increases
- None of them would fit well for DAOS

The Design Requirements for DAMON

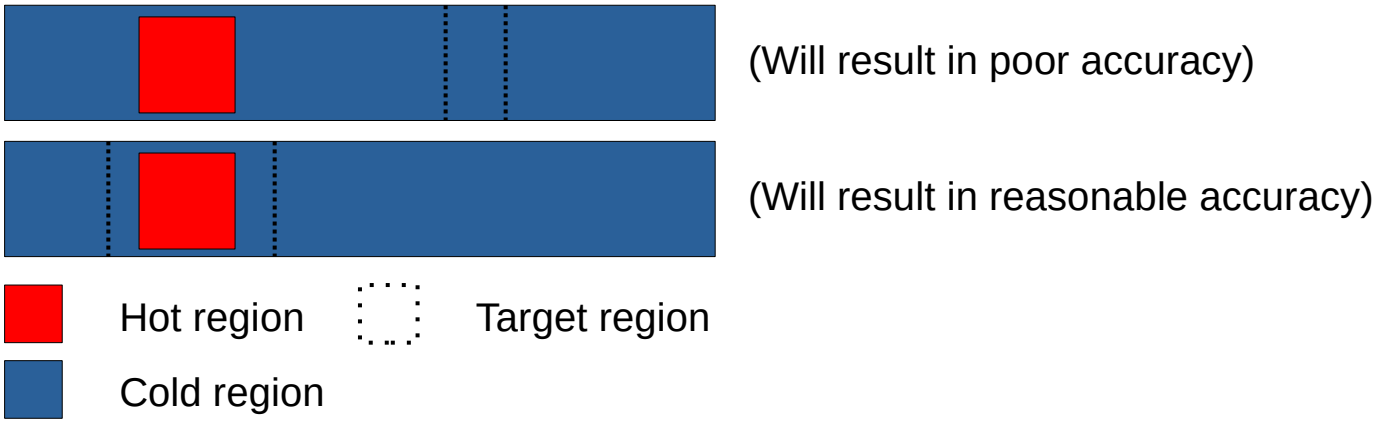
- To be used for DAOS, DAMON needs to fulfill below requirements
  - Accuracy: The monitoring result should be useful for DRAM level MM
  - Overhead: Should light-weight enough for online monitoring
  - Scalability: The upper-bound overhead should be controllable regardless of the size of the monitoring target systems and workloads
- DAMON fulfills the requirements in below steps
  - Straightforward Access Monitoring (collect basic information)
  - Region-based Sampling (optimize overhead)
  - Adaptive Regions Adjustment (make best-effort accuracy)

Straightforward Access Monitoring

- Periodically checks if each monitoring target page is accessed
  - The period is called ‘sampling interval’
- Aggregates the observations into access frequencies
  - Count number of observed accesses and periodically reset the counter
  - The period is called ‘aggregation interval’
- By notifying the users just before the reset of the counter, we can provide the access frequency of the pages to the users
- Pros: Fine-grained (page size) monitoring
  - Might not strictly required in some performance-centric optimizations
- Cons: High and unscalable monitoring overhead
  - The overhead arbitrarily increases as the target size grows

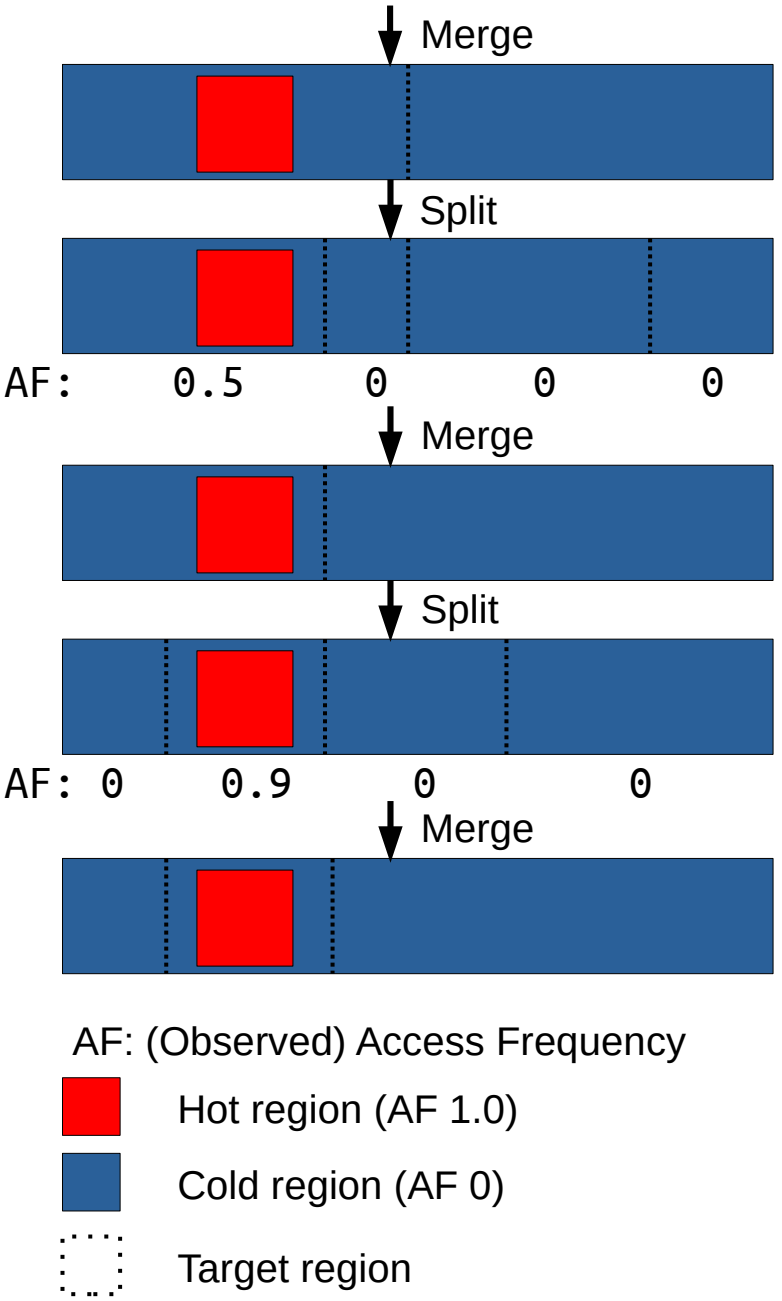
Region-based Sampling

- 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



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



DAMON-based Manual Memory Management Schemes

- Now DAMON-based optimizations are available in below steps
  - Step 1: Run DAMON
  - Step 2: Analyze the monitoring results offline or online
  - Step 3: Make some changes in the kernel or the app
    - e.g., page out cold regions, THP-collapse hot regions, …
  - Could be done inside kernel or from user space
  - Could be effective, but has some risks

Risks of Manual DAMON-based Optimizations

- It could be difficult, dangerous, dirty, and/or restrictive
- Optimizations in the kernel space could be difficult and dangerous
  - No every sysadmin is an experienced kernel programmer
  - Every kernel bug is dangerous
- Optimizations in the user space could be dirty and restrictive
  - Receiving and analyzing the results require some lines of code
    - It also incur overheads for context switches
  - User space apps can do only limited memory management actions
- Most of the basic works would be repetitive

DAMOS: DAMON-based Operation Schemes

- DAMON-based memory management schemes engine for DAOS
- Receives ‘schemes’; each scheme is constructed with
  - Target access pattern: ranges of size, access frequency, and age
  - 1 memory management action
    - Currently supported actions include:
      `WILLNEED, COLD, PAGEOUT, HUGEPAGE, NOHUGEPAGE`
- DAMOS automatically finds the memory region of the target pattern from DAMON results and applies the action to the region
- Now users can make DAMON-based optimizations with no-code

```
# format is:
# <min/max size> <min/max frequency (0-100)> <min/max age> <action>
#
# if a region of size >=4KB didn’t accessed for >=2mins, page out
4K max       0 0       2m max       pageout
```

DAMOS: Example Schemes

- Core concepts of some state-of-the-art works can be implemented using DAMOS again as simple as below
- Proactive reclamation (software-defined far memory, ASPLOS'19)

```
$ cat prcl.damos
# page out memory regions that not accessed >=2 minutes
4K  max     0 0      2m max     page_out
```

- Data Access-aware THP (Ingens, OSDI'16)

```
$ cat ethp.damos
# Use THP for >=2MiB regions having >=80% frequency ratio for >=1 minute
2MB max     80% max     min max     thp
# Don't use THP for regions having <=5% frequency ratio for >=1 minute
min max     min 5%      1m max      nothp
```

Evaluation Questions for DAMON/DAMOS

- How lightweight DAMON is?
- How accurate DAMON is?
- How useful DAMOS-based optimizations are?

Evaluation Environment

- Test machine
  - QEMU/KVM virtual machine on AWS EC2 i3.metal instance
  - 3.0 GHz x 36 vCPUs, 128 GB memory, 4 GB zram swap device
  - Ubuntu 18.04, THP turned off by default
  - Linux v5.10 + DAOS changes
- Workloads: 25 realistic benchmark workloads
  - 13 workloads from PARSEC3
  - 12 workloads from SPLASH-2X
- DAMON monitoring attributes: The default values
  - sampling interval: 5ms, aggregation interval: 100ms
  - Number of regions: [10, 1000]

Evaluation Targets

- Six variants
  - orig: All DAOS features turned off (same to vanilla v5.10 Linux)
  - rec: orig + DAMON for virtual address of the workload turned on
  - prec: orig + DAMON for entire physical address of the system turned on
  - thp: orig + THP enabled with `always` policy (to be compared with ethp)
  - ethp: orig + ethp DAMON-based operation scheme is applied

```
$ cat ethp.damos
# for regions having 5/100 access frequency, apply MADV_HUGEPAGE
min max     5 max       min max     hugepage
# for regions >=2MB and not accessed for >=7 seconds, apply MADV_NOHUGEPAGE
2M max      min min     7s max      nohugepage
```

  - prcl: orig + prcl DAMON-based operation scheme is applied

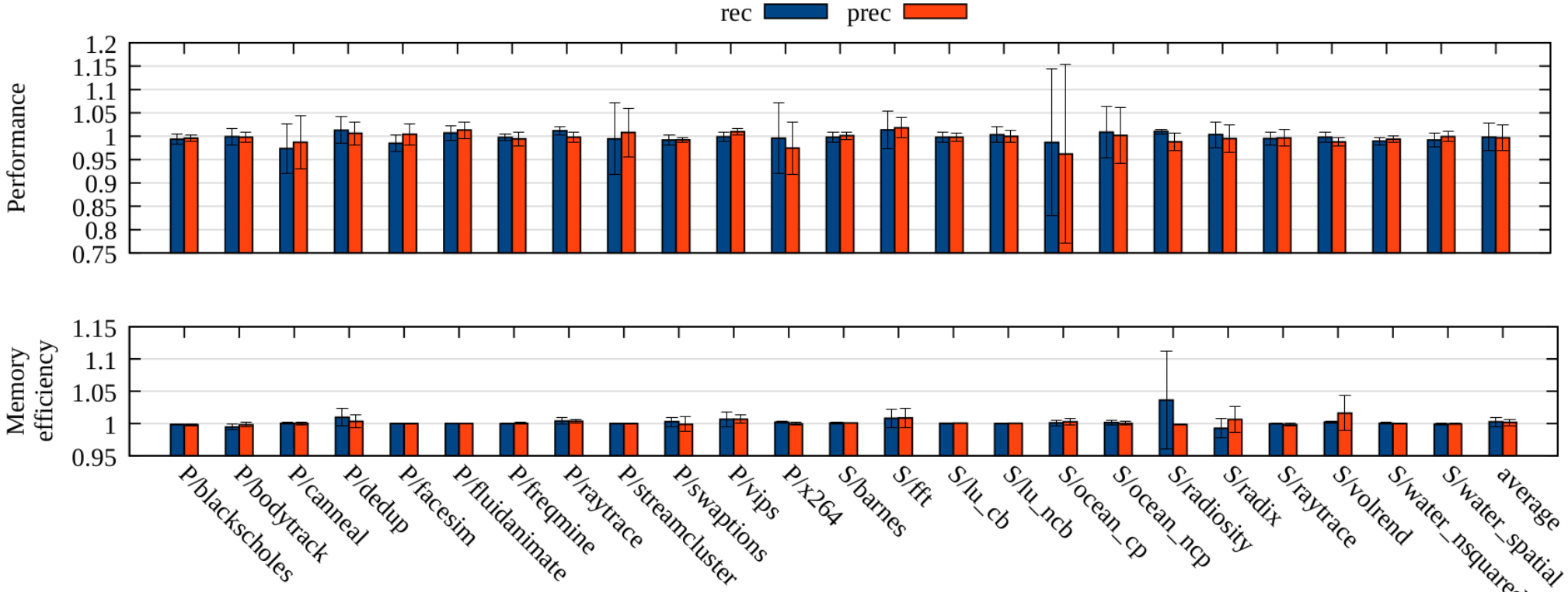
```
$ cat prcl.damos
# for regions >=4KB and not accessed for >=5 seconds, apply MADV_PAGEOUT
4K  max     0 0     5s max      pageout
```

Evaluation Methodology

- Measurement
  - DAMON/DAMOS' CPU consumption
  - Runtime speedup compared to the 'orig'
  - Memory efficiency compared to the 'orig'
    - System entire memory usage for 'prec', RSS of the workload for others
  - For each of the workload x variant combinations
- Every following data is average of 5 different runs

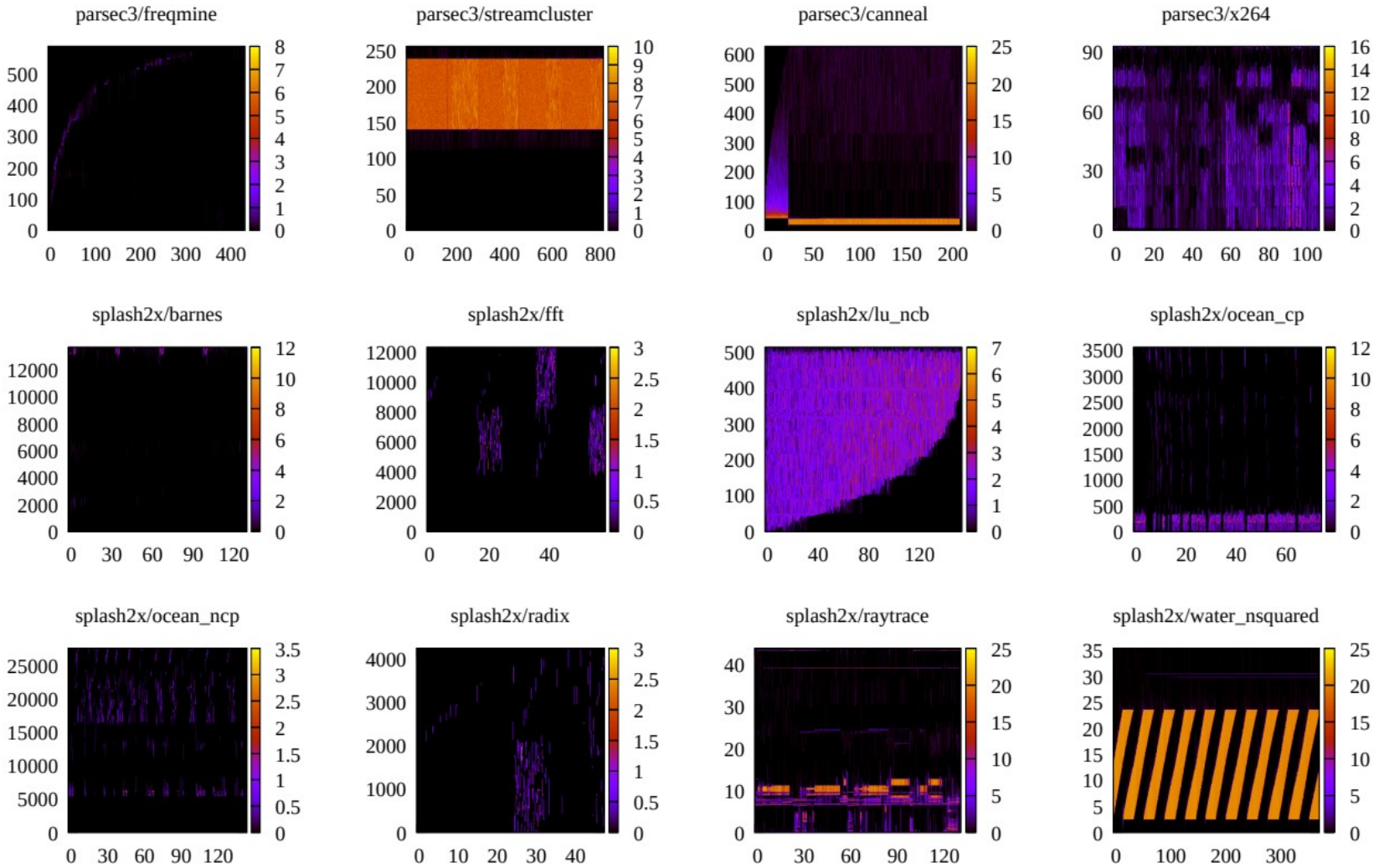
DAMON Overhead

- For both ‘rec’ and ‘prec’ (note: ‘prec’ covers entire system memory)
  - Consumes <2% of a single CPU time
  - Incurs about 1% on average and up to 4% slowdown to the workload
  - Memory overhead is also negligible
- DAMON keeps upper-limit of the overhead regardless of target size



DAMON Accuracy

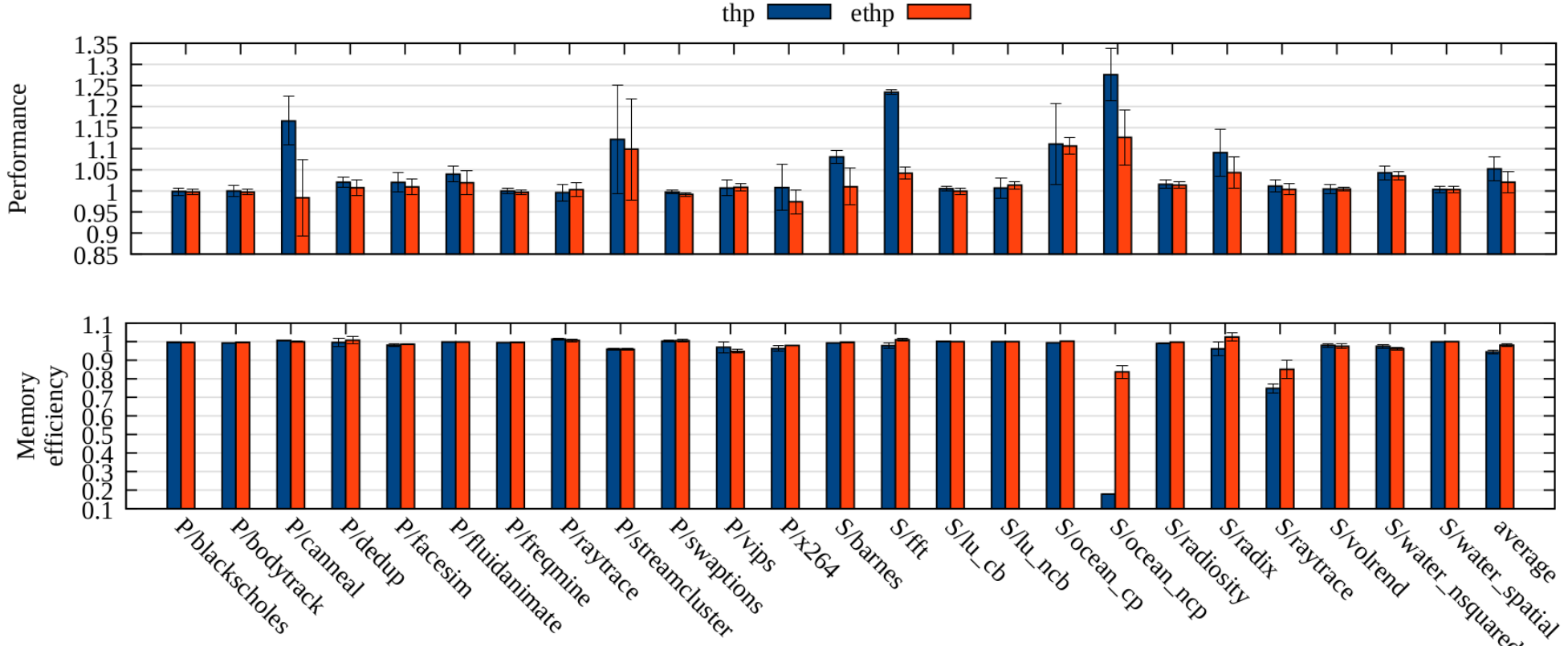
- DAMON results in heatmap visualization shows reasonable results



Data access patterns of the workloads in heatmap format. Each heatmap shows when (x-axis, in seconds) what memory regions of specific address (y-axis, in MiB) is how frequently (color) accessed.

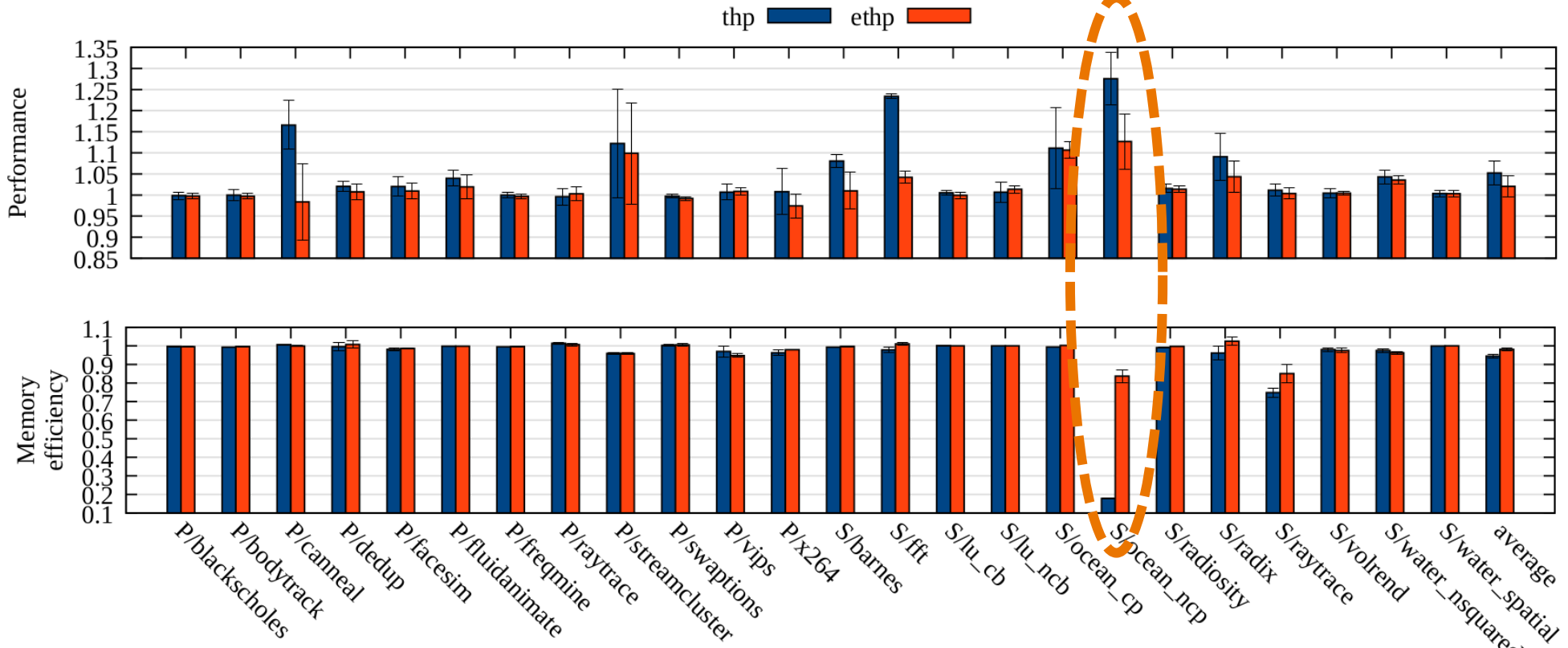
DAMON Accuracy and DAMOS ‘ethp’ Effectiveness

- On average, ‘ethp’ preserves 39% of ‘thp’ speedup while removing 64.28% of ‘thp’ memory overhead
- For splash-2x/ocean_ncp
  - ‘thp’ shows up to 27.54% speedup with 82.18% memory overhead
  - ‘ethp’ provides 12.67% speedup with only 16.3% memory overhead



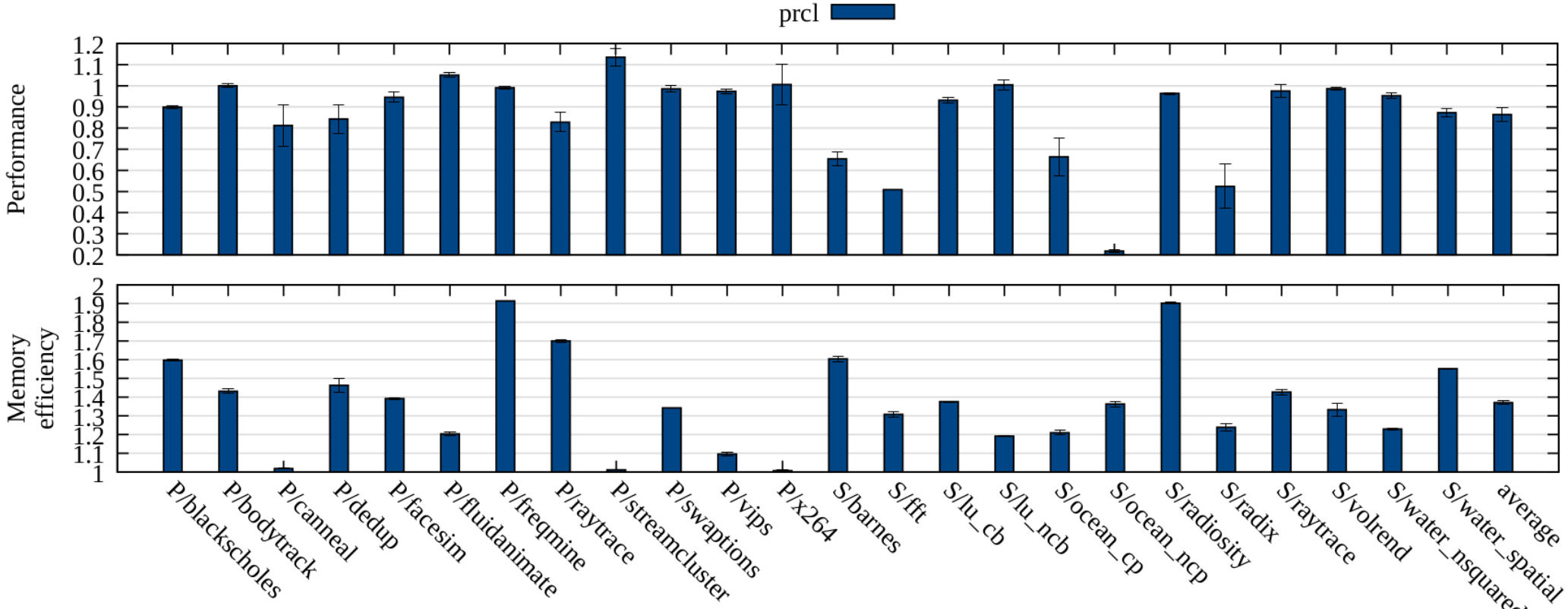
DAMON Accuracy and DAMOS ‘ethp’ Effectiveness

- On average, ‘ethp’ preserves 39% of ‘thp’ speedup while removing 64.28% of ‘thp’ memory overhead
- For splash-2x/ocean_ncp
  - ‘thp’ shows up to 27.54% speedup with 82.18% memory overhead
  - ‘ethp’ provides 12.67% speedup with only 16.3% memory overhead



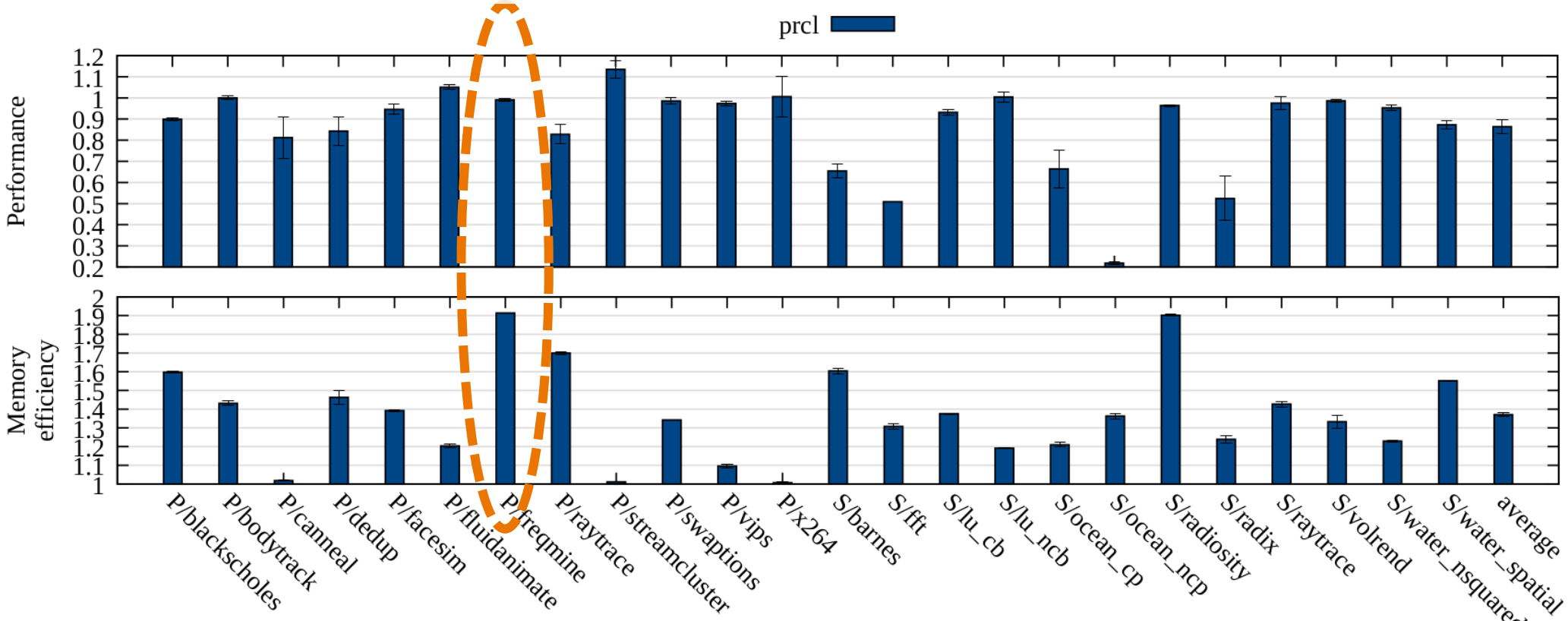
DAMON Accuracy and DAMOS ‘prcl’ Effectiveness

- On average, ‘prcl’ saves 37.10% memory with 13.66% slowdown
  - On the best case (‘parsec3/freqmine’), 91.34% memory saving with 0.91% slowdown
  - On the worst case (‘splash-2x/ocean_ncp’), 36.29% memory saving with 78.16% slowdown; coldness threshold would need to be longer



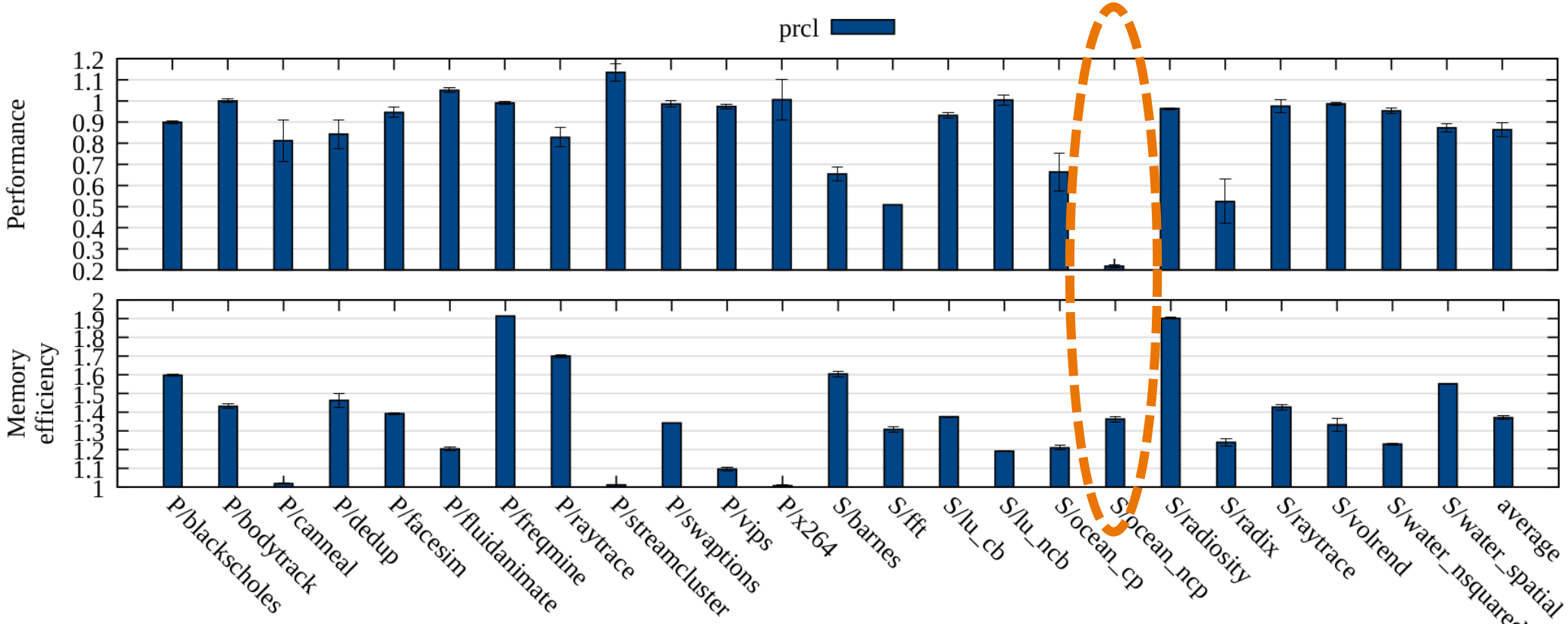
DAMON Accuracy and DAMOS ‘prcl’ Effectiveness

- On average, ‘prcl’ saves 37.10% memory with 13.66% slowdown
  - On the best case (‘parsec3/freqmine’), 91.34% memory saving with 0.91% slowdown
  - On the worst case (‘splash-2x/ocean_ncp’), 36.29% memory saving with 78.16% slowdown; coldness threshold would need to be longer



DAMON Accuracy and DAMOS ‘prcl’ Effectiveness

- On average, ‘prcl’ saves 37.10% memory with 13.66% slowdown
  - On the best case (‘parsec3/freqmine’), 91.34% memory saving with 0.91% slowdown
  - On the worst case (‘splash-2x/ocean_ncp’), 36.29% memory saving with 78.16% slowdown; coldness threshold would need to be longer



Conclusion from DAMON/DAMOS Evaluation

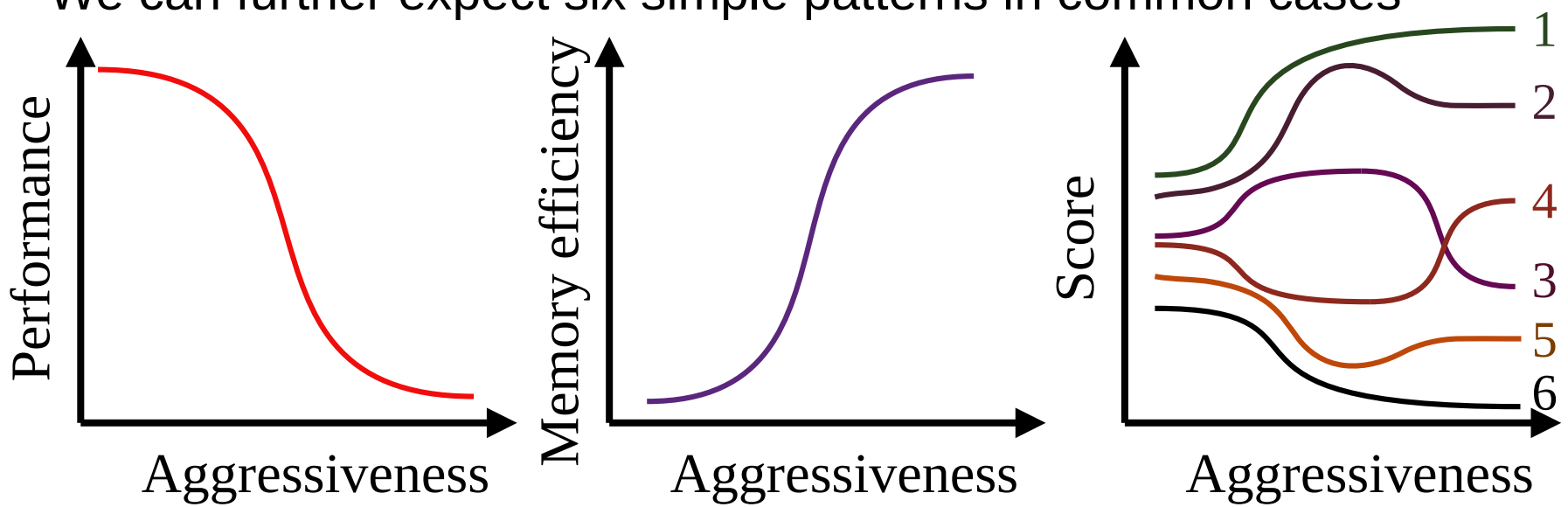
- DAMON is lightweight, scalable, and accurate for DRAM level optimization
- DAMOS is effective, but needs fine tuning for each workload
  - This would be not trivial works; Should we do that every time?

Why Searching Optimal Operation Schemes Is Difficult

- Depends on both h/w and s/w characteristics
  - Need to tune for each application on each system
- Many parameters to consider and tune
  - TLB miss, pgmajfaults, CPU usage, swap i/o, PSI, …
    - Target access pattern itself is constructed with six data points
  - Essentially this problem is multi-dimensional search problem

Simplifying The Problem

- We care only memory efficiency and performance at last
  - These can be consolidated into one metric (score) with different priorities
  - Giving the metrics and priorities could be easy for users (like SLO)
- The target access pattern is only the aggressiveness of the scheme
- The multi-dimension search space can be reduced to 2-dimension
  - Aggressiveness as X-axis, Score as Y-axis
- We can further expect six simple patterns in common cases



DAMOOS: Auto-tuning Runtime for DAOS

- Receives
  - A workload to run
  - Methods for measuring the performance and memory efficiency
  - The score function for the workload
  - Unit work time of the workload
    - The time to wait before getting performance and memory efficiency
  - Time limit for the tuning
- Finds the best DAMOS scheme for the score function in the time limit and starts the workload with the scheme

DAMOOS: Sampling (Example: ‘prcl’)

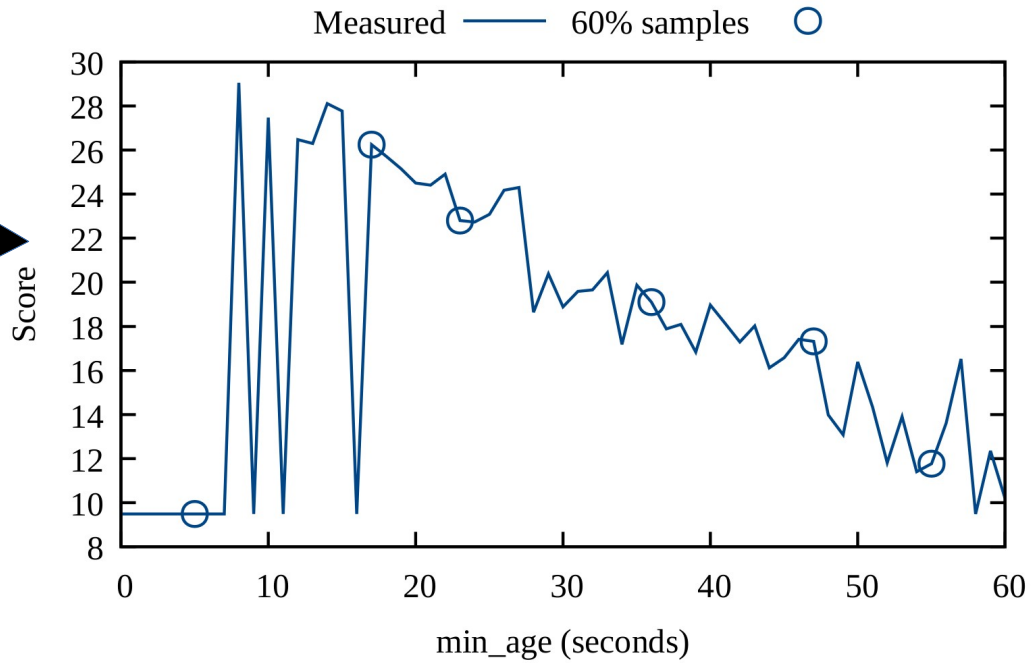
- Calculate how many times we can measure the score for different aggressiveness (‘nr_samples’)
  - The user-specified tuning time limit divided by the unit work time
- Run the workload with 60% of ‘nr_samples’ schemes having random aggressiveness and measure one score for each scheme
- Run the workload with 40% of ‘nr_samples’ schemes having random but near to the best of the 60% sample results aggressiveness

‘prcl’ example. Measured line is what we measured later (DAMOOS don’t know it)

DAMOOS: Sampling (Example: ‘prcl’)

- Calculate how many times we can measure the score for different aggressiveness (‘nr_samples’)
  - The user-specified tuning time limit divided by the unit work time
- Run the workload with 60% of ‘nr_samples’ schemes having random aggressiveness and measure one score for each scheme
- Run the workload with 40% of ‘nr_samples’ schemes having random but near to the best of the 60% sample results aggressiveness

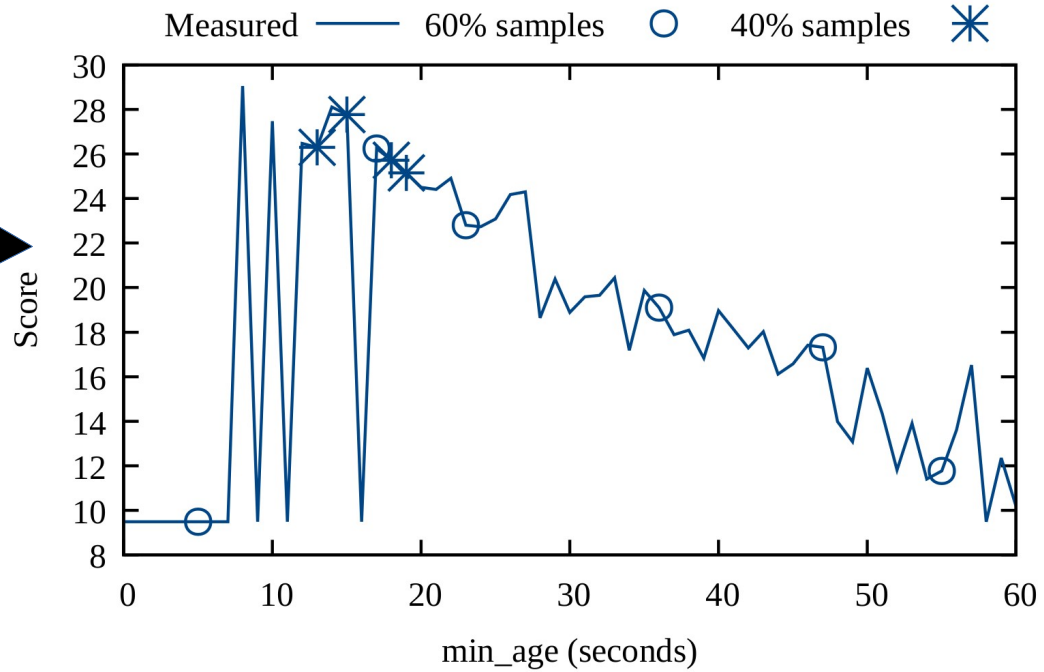
‘prcl’ example. Measured line is what we measured later (DAMOOS don’t know it)



DAMOOS: Sampling (Example: ‘prcl’)

- Calculate how many times we can measure the score for different aggressiveness (‘nr_samples’)
  - The user-specified tuning time limit divided by the unit work time
- Run the workload with 60% of ‘nr_samples’ schemes having random aggressiveness and measure one score for each scheme
- Run the workload with 40% of ‘nr_samples’ schemes having random but near to the best of the 60% sample results aggressiveness

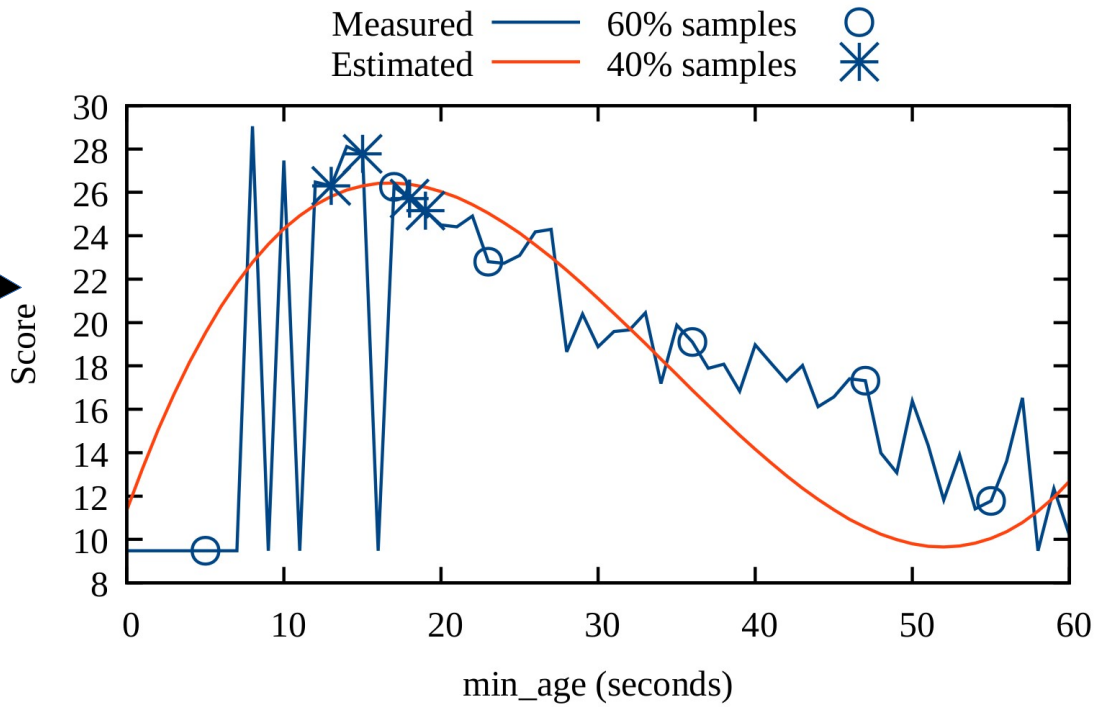
‘prcl’ example. Measured line is what we measured later (DAMOOS don’t know it)



DAMOOS: Estimation and Best Scheme Selection

- Find the relationship between the aggressiveness and score by applying Polynomial curve fitting to the ‘nr_samples’ data points
- On the curve, we find an aggressiveness value that generates maximum score and use it as the best scheme aggressiveness

‘prcl’ example. Measured line is what we measured later (DAMOOS don’t know it)

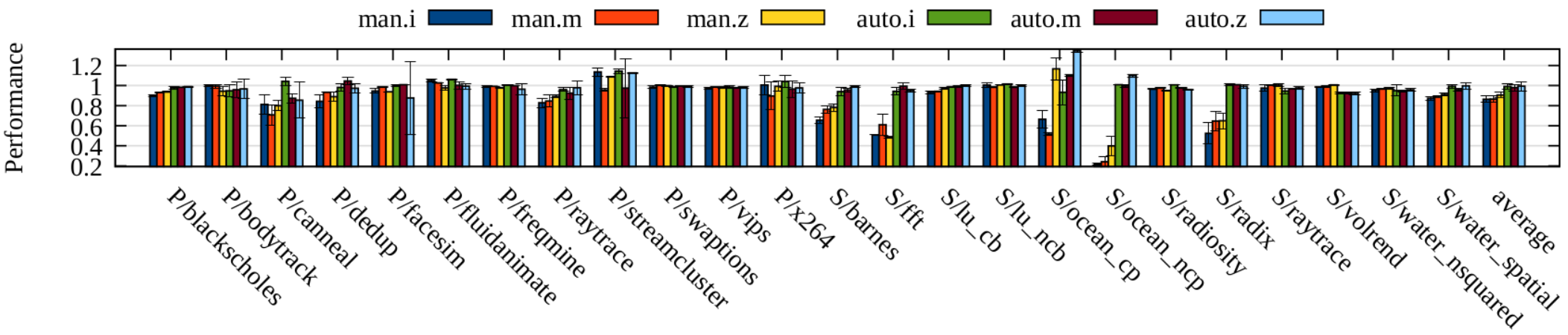


DAMOOS Evaluation Setup

- Basically same to the DAMON/DAMOS evaluation setup
- Adds two more AWS EC2 instance types to show if DAMOOS works for different hardware types
- In total, three types of hardware are used
  - I3.metal: 3 GHz x 36 vCPUs, 128 GiB DRAM (i/o optimized)
  - M5d.metal: 3.1 GHz x 48 vCPUs, 96 GiB DRAM (general purpose optimized)
  - Z1d.metal: 4.0 GHz x 24 vCPUs, 96 GiB DRAM (computing optimized)
- Use ‘prcl’ only
- The score function is focused on avoiding >10% performance drop
  - For detail, please read the paper

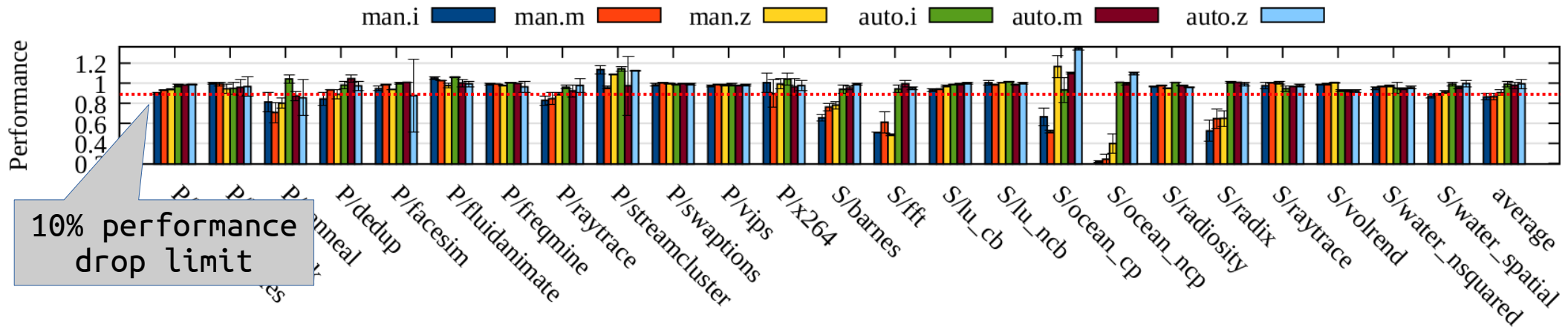
DAMOOS Evaluation Results for ‘prcl’: Performance

- Average performance drops on the three h/w
  - Manual-tuned: 13.65%, 13.45%, 9.54%
  - Auto-tuned: 0.91%, 2.04%, 0.59%
- DAOS’s auto-tuning removes 93.33%, 84.83%, and 93.81% of the slowdown of manual version on the three h/w, respectively
  - The 10% performance drop SLO is almost always kept



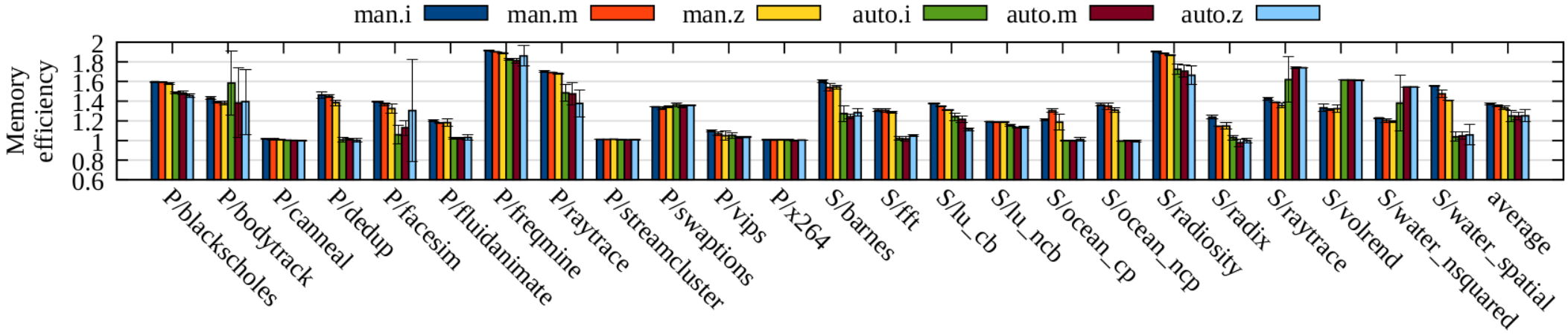
DAMOOS Evaluation Results for ‘prcl’: Performance

- Average performance drops on the three h/w
  - Manual-tuned: 13.65%, 13.45%, 9.54%
  - Auto-tuned: 0.91%, 2.04%, 0.59%
- DAOS’s auto-tuning removes 93.33%, 84.83%, and 93.81% of the slowdown of manual version on the three h/w, respectively
  - The 10% performance drop SLO is almost always kept



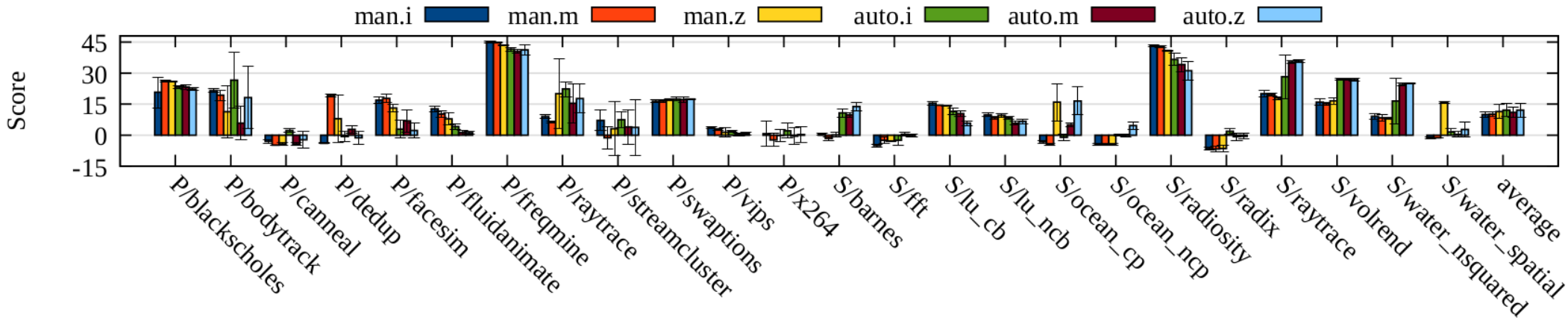
DAMOOS Evaluation Results for ‘prcl’: Memory Saving

- Average memory saving on the three h/w types
  - Manually-tuned: 37.10%, 35.21%, 33.22%
  - Auto-tuned: 24.97%, 24.73%, 25.10%
- Auto-tuned version’s memory saving is lower than those of manual version
  - Because the score function gives more priority to performance
  - Memory saving could increase if we adjust the priorities



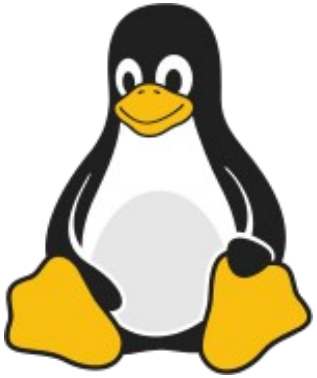
DAMOOS Evaluation Results for ‘prcl’: Score

- Average score on the three h/w types
  - Manually-tuned: 9.99, 10.09, 11.35
  - Auto-tuned: 12.08, 11.01, 12.06
- Auto-tuning runtime obtains score improvements of 20.02%, 6.16%, and 6.25%



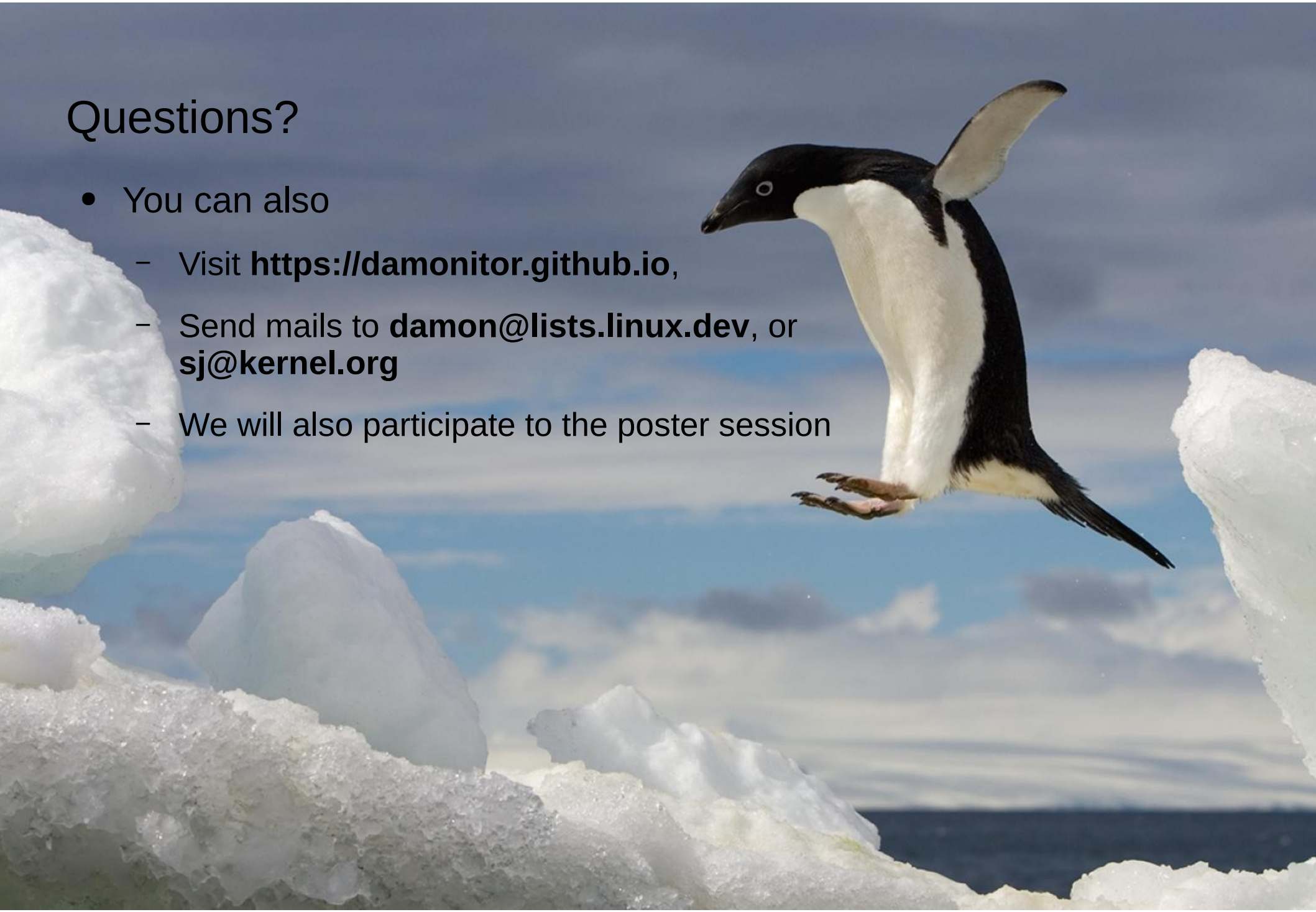
Conclusion

- DAOS is a data access-aware operating system that constructed with three simple pieces
  - DAMON provides best-effort accurate access patterns with lightweight and upper-bound-limited overhead
  - DAMOS allows general access-aware memory management with no difficult, danger, dirty code
  - DAMOOS provides a runtime that provides auto-tuned data access-aware memory management scheme within a finite time
- All parts of DAOS are opensource
  - DAMON and DAMOS are merged in the mainline Linux kernel
  - DAMOOS is available at github
  - Visit https://github.com/damonitor for quick start



Questions?

- You can also
  - Visit **https://damonitor.github.io**,
  - Send mails to **damon@lists.linux.dev**, or **sj@kernel.org**
  - We will also participate to the poster session



https://kids.nationalgeographic.com/content/dam/kids/photos/animals/Birds/A-G/adelie-penguin-jumping-ocean.ngsversion.1396530997321.adapt.1900.1.jpg

Backup Slides

DAMON: Resulting Architecture

- Core logic and monitoring operations layer are separated
  - Multiple address spaces and usages can easily supported
  - Current version of DAMON supports virtual address spaces and the physical address space