DAMON Updates and Future Plans:

Automation of DAMON tuning, tiering, and VM guest scaling

SeongJae Park <sj@kernel.org>

https://damonitor.github.io

Notices

- The views expressed herein are those of the speaker;
  they do not reflect the views of his employers
- Slides are available at https://github.com/damonitor/talks or below QR code

Scan to download these slides

QR code is generated by https://qr.io/

From: SeongJae Park <sj@kernel.org>

- Just call me “SJ” (easier to be consistently pronounced)
- Interested in the memory management and the parallel programming
- Maintaining DAMON (`mm/damon/`)
- Kernel Development Engineer at AWS

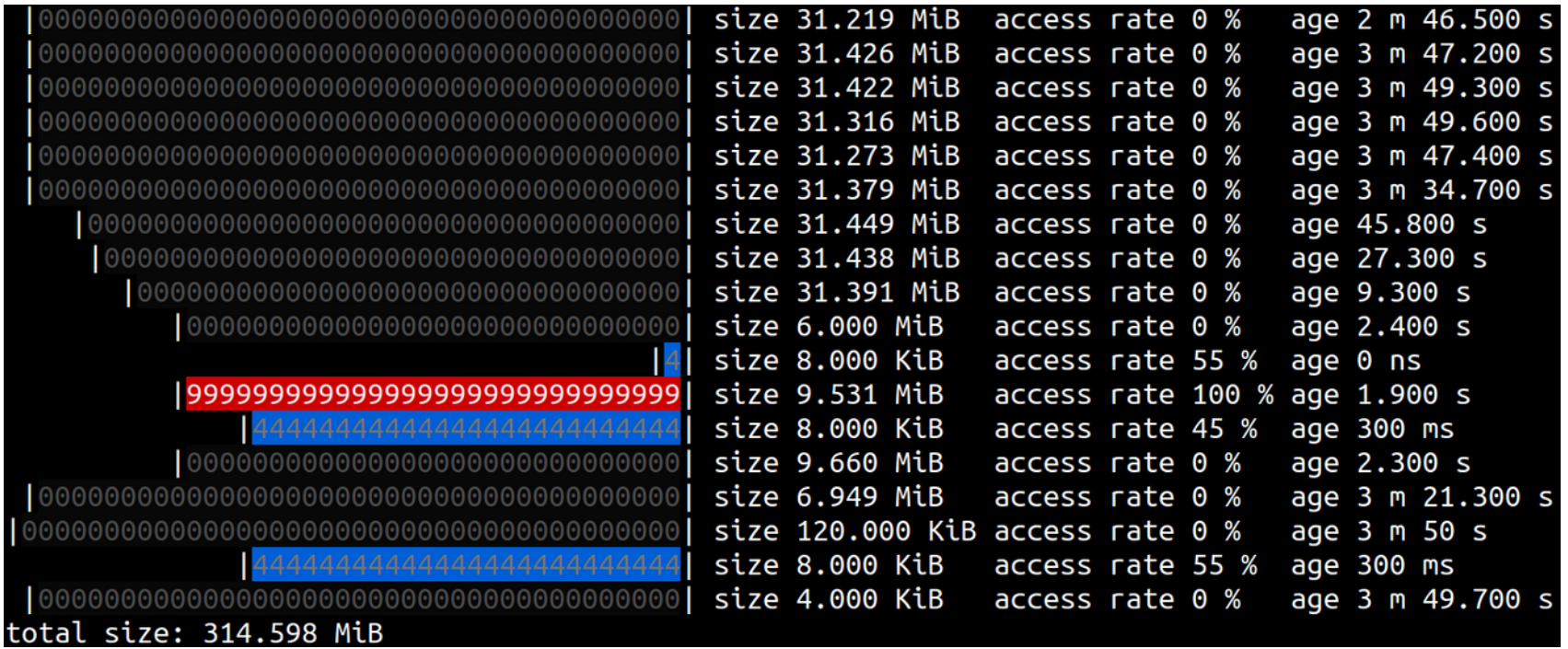
Overview

- DAMON in a Nutshell (2 mins)
- Updates since LSFMM+BPF 2023 (5 mins)
- Major Future Plans (13 mins)
  - Tiered Memory Management (4 mins)
  - Access/Contiguity-aware Memory Auto-scaling (6 mins)
  - Misc Plans (3 mins)
- Discussions (10 mins)

DAMON in a Nutshell

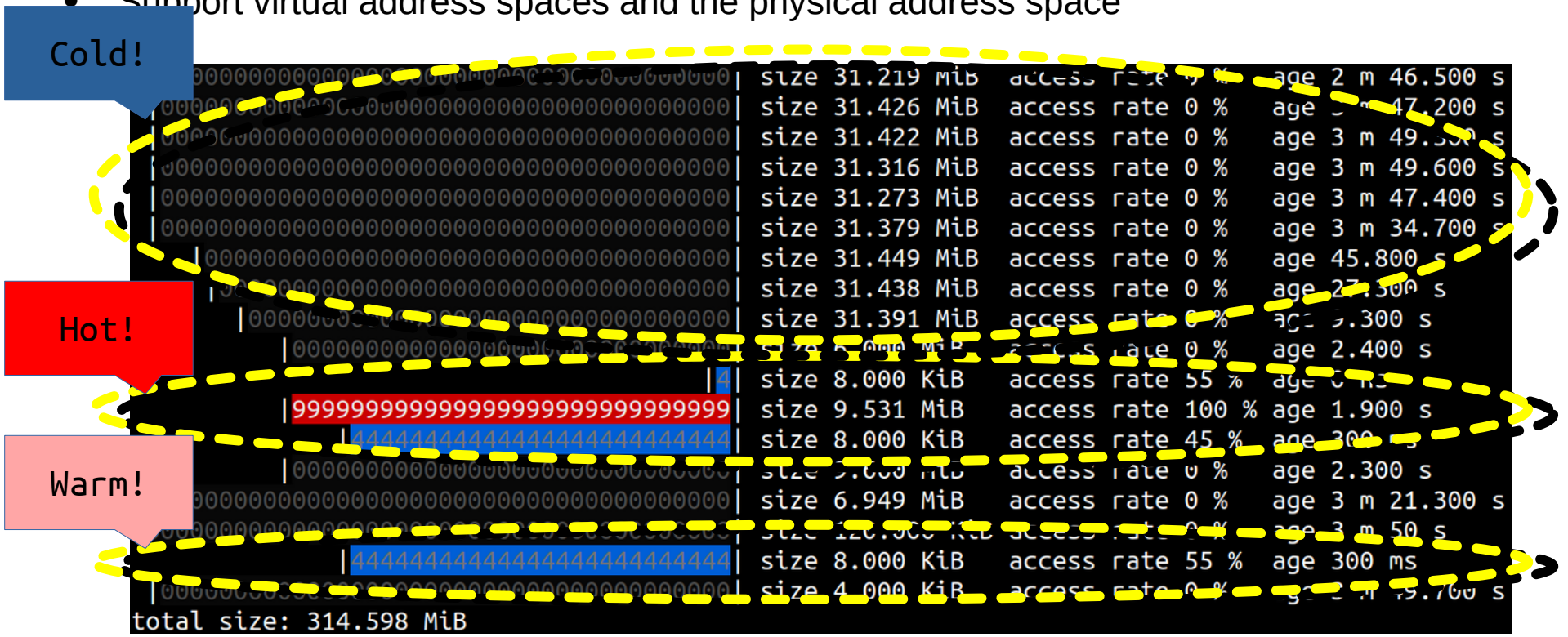
DAMON: Access Pattern Snapshot Generator

- Inform which *address range* is how *frequently* accessed for how *long* time
- Support virtual address spaces and the physical address space



DAMON: Access Pattern Snapshot Generator

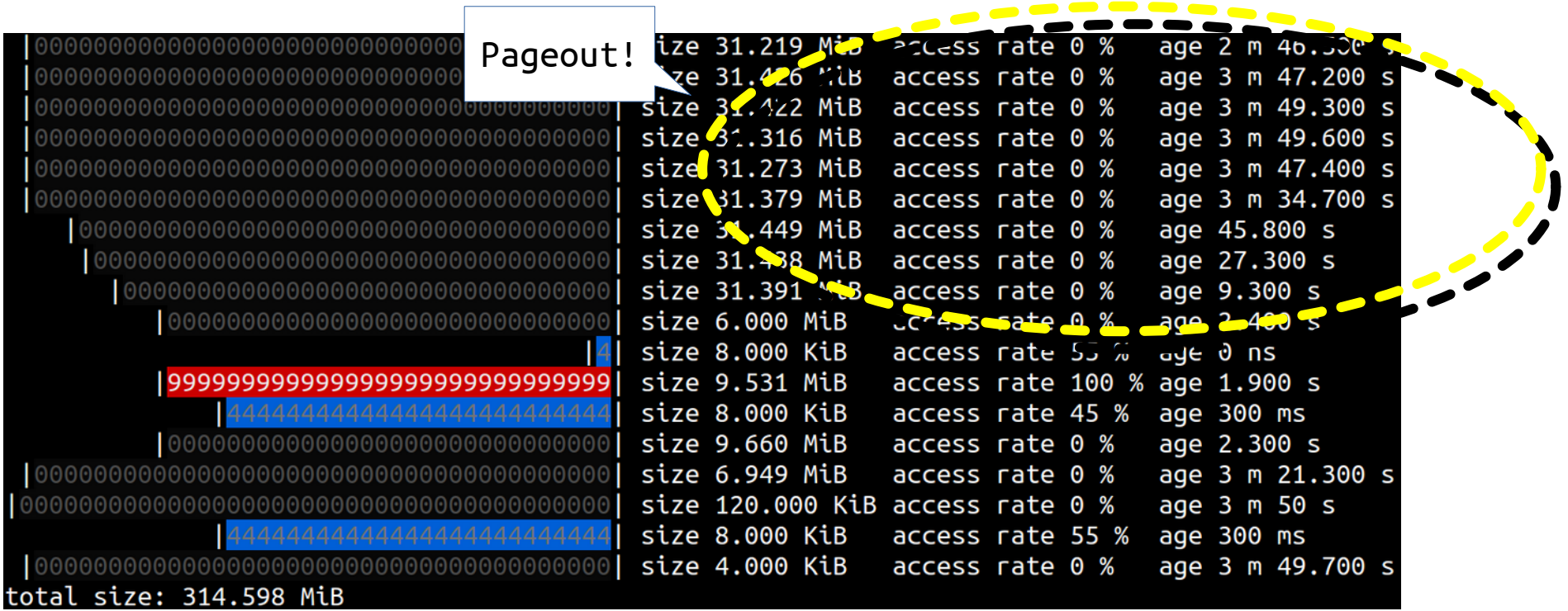
- Inform which *address range* is how *frequently* accessed for how *long* time
- Support virtual address spaces and the physical address space



DAMOS: DAMON-based Operation Scheme

- Apply memory operation actions to regions of interesting access pattern

```
# # pageout memory regions that not accessed for >=5 seconds
# damo start --damos_action pageout --damos_access_rate 0% 0% --damos_age 5s max
```



Features for Product Quality DAMOS Control

- *“One does not simply control DAMOS with only access pattern”*
- Quota: set aggressiveness of DAMOS
  - e.g., pageout cold pages up to 100 MiB per second (coldest 100 MiB pages)
- Filters: define target regions with non-access-pattern information
  - e.g., pageout cold pages of NUMA node 1 that associated with cgroup “A” and file-backed

Usages, To One’s Best Knowledge (No Way to Know Every Usage)

- Products
  - Proactive memory reclamation on memory overcommit systems
  - CXL-based tiered memory management software development kit
- Researches
  - DAMON paper got 20 citations
- Distros having `CONFIG_DAMON=y`
  - Amazon Linux (>=5.4), Android (>=5.10), CentOS (>=4.18), Fedora (>=6.2), UEK (>=5.15)
- Package repos providing DAMON user-space tool
  - AUR, Debian, EPEL, Fedora, Kali, PyPI, Raspian, Ubuntu

Community

- *“Strive to be Earth’s best community”*
- Public mailing list (https://lore.kernel.org/damon)
- Bi-weekly virtual meetup
- Occasional/regular private meetings on demand
- Project website (https://damonitor.github.io)
  - Starting point for DAMON users and developers
  - Daily performance test results archive

Scan to visit the project website

QR code is generated by https://qr.io/

DAMON Updates

Since LSFMM+BPF 2023

Answer to LSFMM 2023 Feedbacks

- “Some good documentation would be appreciated”
  - Improving `Documentation/.../damon/`
- “Adding DAMON user-space tool in-tree sounds not a good idea”
  - The goal was test coverage and easy DAMON interface understanding
  - Implementing DAMON functionality selftests

```
v6.5-rc1
    Patch series "Docs/{mm,admin-guide}damon: update design and usage docs".
    Patch series "Docs/mm/damon: Minor fixes and design doc update".
v6.8-rc1
    Patch series "selftests/damon: add Python-written DAMON functionality
v6.9-rc1
    Patch series "Docs/mm/damon: misc readability improvements".
    Patch series "selftests/damon: add more tests for core functionalities and
mm-stable
    Patch series "mm/damon: misc fixes and improvements".
    Patch series "selftests/damon: add DAMOS quota goal test".
```

Pseudo-moving Sum-based Fast Snapshot Generation

- Complete access pattern snapshot is generated per “aggregation interva”
  - “Aggregation interval”: 100ms by default and user-tunable
  - Large “aggregation interval” (say, 20 seconds) limits monitoring and DAMOS
- Generate reasonable-quality snapshot per “sampling interval”
  - “Sampling interval”: 5ms by default and user-tunable
  - Monitoring and DAMOS per < “aggregation interval” available

```
v6.7-rc1
    Patch series "mm/damon: implement DAMOS apply intervals".
    Patch series "mm/damon: provide pseudo-moving sum based access rate".
```

New Filter Types

- Address Range
  - For applying DAMOS to specific NUMA nodes, zones, virtual memory area, etc
- Young Page
  - For page granularity access double check
  - Complement DAMON’s best-effort accuracy

```
v6.6-rc1
    Patch series "Extend DAMOS filters for address ranges and DAMON monitoring
mm-stable
    Patch series "mm/damon: add a DAMOS filter type for page granularity
```

Aim-oriented Feedback-driven DAMOS Aggressiveness Auto-tuning

- *“One does not manually control DAMOS”*
- Auto-tune effective DAMOS quota using a proportional feedback loop
- Two ways to provide feedback
  - User-providing arbitrary value (e.g., Main workload’s latency)
    - *“Good DAMOS! … Bad DAMOS! … Bad Bad DAMOS!!”*
  - System metrics and their target value (e.g., 0.1% memory PSI)
    - Users set the target once; DAMOS self-feed/auto-tune
    - Supporting memory PSI-based target (will add more target metrics)
    - *“Reclaim cold pages aiming 0.1% memory pressure stall rate”*

```
v6.8-rc1
    Patch series "mm/damon: let users feed and tame/auto-tune DAMOS".
v6.9-rc1
    Patch series "mm/damon: let DAMOS feeds and tame/auto-tune itself".
```

DAMON Future Plans

DAMOS Auto-tuning Based Tiered Memory Management

Existing DAMON-based Tiered Memory Management Approaches

- MTM: Multi-Tiered Memory Management (Jie Ren et al., Eurosys'24)
  - Propose DAMON-feasible monitoring improvements
- Two-tier memory promotion/demotion (HMSDK v2, SK hynix)
  - Migrate hot/cold pages to upper/lower tier using DAMOS
  - Patchset is available (actively working, merged in `damon/next` tree)
  - Motivated '`young page`' type DAMOS filter
  - Patches implement only mechanisms, not the policy

DAMOS-based Tiered Memory Management Policy Proposal

- For each CPU-independent NUMA node,
  - If the node has a lower node,
    - Demote cold pages of the current node to the lower node,
      aiming little fraction (e.g. 5%) of free memory of the current node
  - If the node has a upper node,
    - Promote hot pages of the current node to the upper node,
      aiming big fraction (e.g., 96%) of used memory of the _upper_ node

```
node 0 (fast)  Demote cold pages in node 0 aiming 5% free memory of node 0
node 1 (slow)  Promote hot pages in node 1 aiming 96% used memory of node 0
               Demote cold pages in node 1 aiming 5% free memory of node 1
node 2 (slowoo)Promote hot pages in node 2 aiming 96% used memory of node 1
```

Expectations, or Hopes

- High utilization of upper nodes, with more frequently accessed pages
- Low utilization of lower nodes, with less frequently accessed pages
- Keep slow but continuous promotion/demotion
  - Overlapping memory util/free goals
- Easy to be extended for multiple tiers
- Take some time to reach out to the final destination tier (bubbling up/down)
  - Depending on workloads and time scale, no problem

Progress

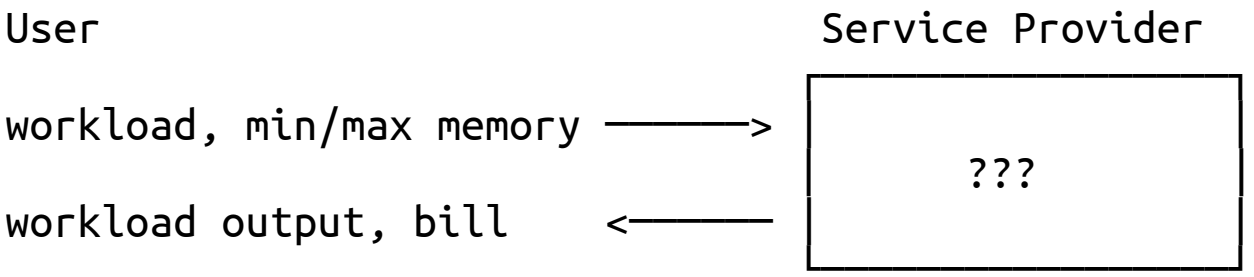
- Detailed RFC idea is sent to the mailing list
- No test setup, no implementation

Access/Contiguity-aware Memory Auto-scaling (ACMA)

https://lore.kernel.org/20240512193657.79298-1-sj@kernel.org

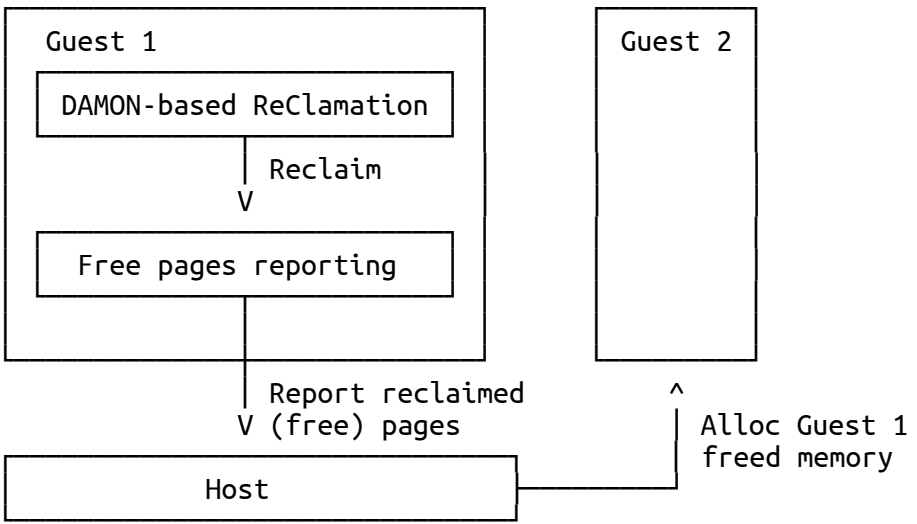
Motive Business Model

- User: specify workload and min/max memory requirements
- Service Provider: run it somewhere, charge as they gone
  - Achieving high performance and low price is the provider’s duty, and benefits both



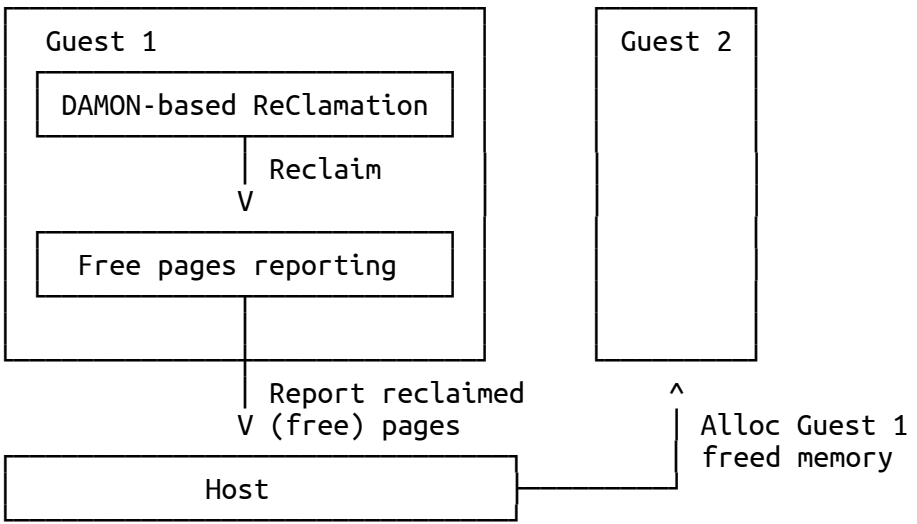
An Existing Approach: Orchestration of Four Kernel Features

- Collaborative overcommit (Free pages reporting)
- DAMON_RECLAIM for reporting more pages without performance degradation
- Periodic compaction for reporting level contiguity
- Memory hot-[un]plugging for hard limit and ‘struct page’ reduction
- Works well in real world



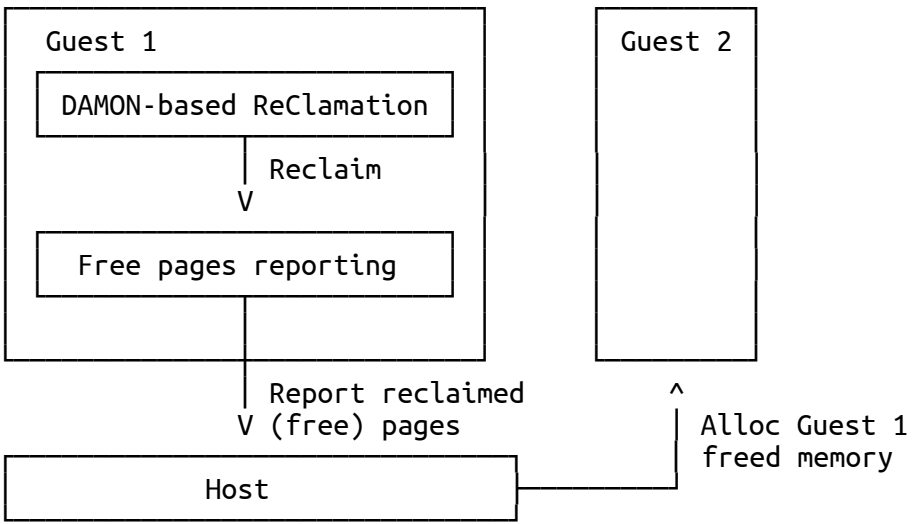
An Existing Approach: Orchestration of Four Kernel Features

- Collaborative overcommit (Free pages reporting)
- DAMON_RECLAIM for reporting more pages without performance degradation
- Periodic compaction for reporting level contiguity
- Memory hot-[un]plugging for hard limit and ‘struct page’ reduction
- Works well in real world



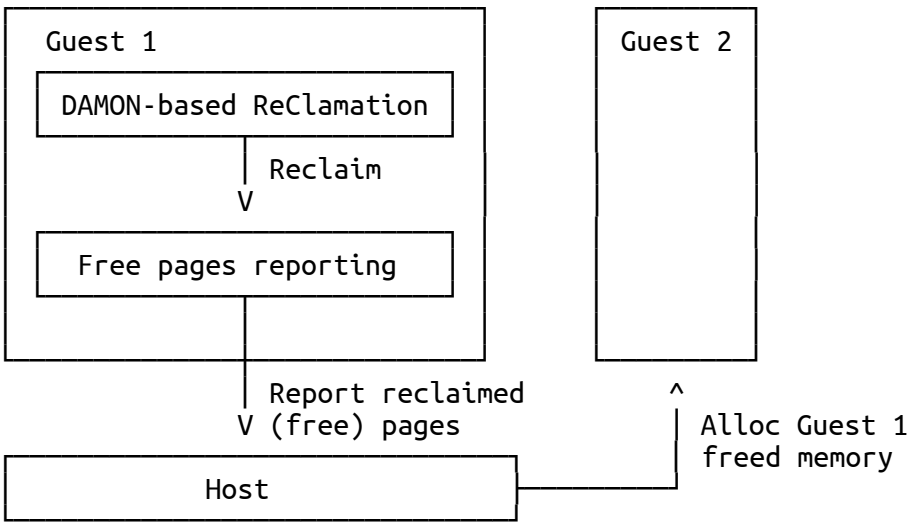
An Existing Approach: Orchestration of Four Kernel Features

- Collaborative overcommit (Free pages reporting)
- DAMON_RECLAIM for reporting more pages without performance degradation
- Periodic compaction for reporting level contiguity
- Memory hot-[un]plugging for hard limit and ‘struct page’ reduction
- Works well in real world



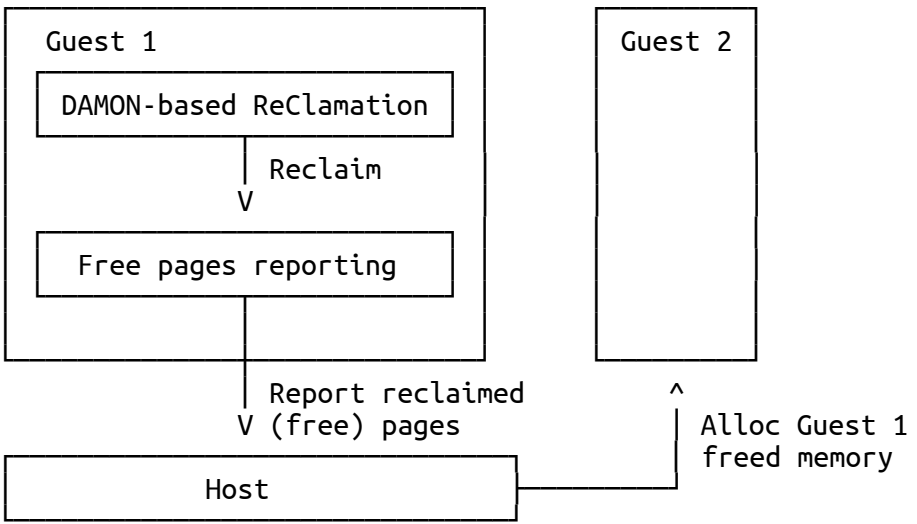
An Existing Approach: Orchestration of Four Kernel Features

- Collaborative overcommit (Free pages reporting)
- DAMON_RECLAIM for reporting more pages without performance degradation
- Periodic compaction for reporting level contiguity
- Memory hot-[un]plugging for hard limit and ‘struct page’ reduction
- Works well in real world



An Existing Approach: Orchestration of Four Kernel Features

- Collaborative overcommit (Free pages reporting)
- DAMON_RECLAIM for reporting more pages without performance degradation
- Periodic compaction for reporting level contiguity
- Memory hot-[un]plugging for hard limit and ‘struct page’ reduction
- Works well in real world



Limitations

- Complexity of user-space driven multiple kernel features orchestration
- Memory hot-unplugging is slow and easy to fail
  - Due to coarse granularity and access obliviousness
- System-level compaction is wasteful and access oblivious
- Lack of after-report pages control
  - Any reported pages can be claimed again at any time
- Lack of non-collaborative guests control

Limitations

- Complexity of user-space driven multiple kernel features orchestration
- Memory hot-unplugging is slow and easy to fail
  - Due to coarse granularity and access obliviousness
- System-level compaction is wasteful and access oblivious
- Lack of after-report pages control
  - Any reported pages can be claimed again at any time
- Lack of non-collaborative guests control

Limitations

- Complexity of user-space driven multiple kernel features orchestration
- Memory hot-unplugging is slow and easy to fail
  - Due to coarse granularity and access obliviousness
- System-level compaction is wasteful and access oblivious
- Lack of after-report pages control
  - Any reported pages can be claimed again at any time
- Lack of non-collaborative guests control

Limitations

- Complexity of user-space driven multiple kernel features orchestration
- Memory hot-unplugging is slow and easy to fail
  - Due to coarse granularity and access obliviousness
- System-level compaction is wasteful and access oblivious
- Lack of after-report pages control
  - Any reported pages can be claimed again at any time
- Lack of non-collaborative guests control

Limitations

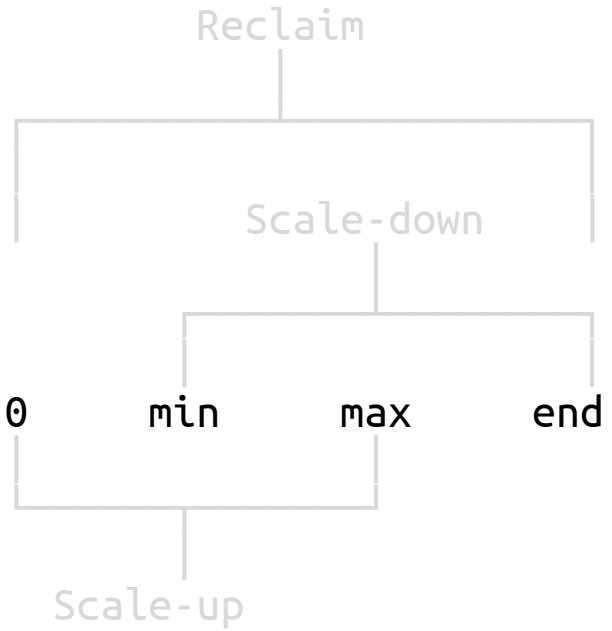
- Complexity of user-space driven multiple kernel features orchestration
- Memory hot-unplugging is slow and easy to fail
  - Due to coarse granularity and access obliviousness
- System-level compaction is wasteful and access oblivious
- Lack of after-report pages control
  - Any reported pages can be claimed again at any time
- Lack of non-collaborative guests control

DAMOS Actions for Access-aware Contiguous Memory Allocation

- `DAMOS_ALLOC`
  - Allocate given memory region with user-specified minimum contiguity
  - Notify (callback) the allocation to the user
  - “Repeatedly try to allocate cold memory regions, 2 MiB contig-regions at once”
- `DAMOS_FREE`
  - De-allocate the region with user-specified minimum contiguity
  - Notify the user to be prepared before de-allocation

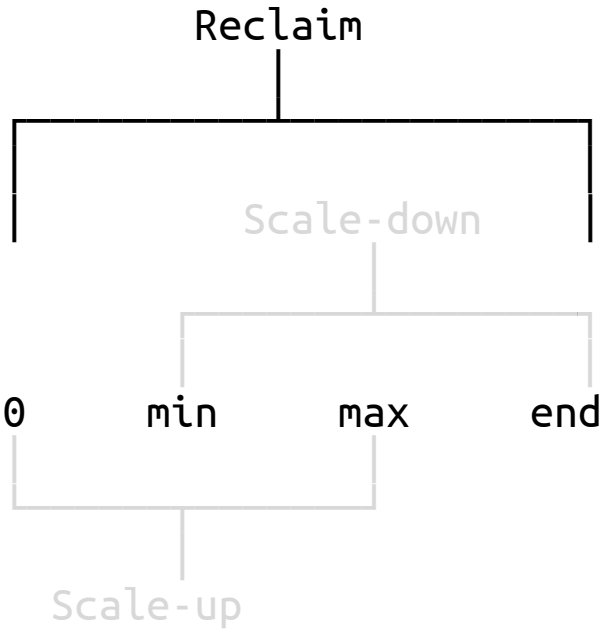
Access/Contiguity-aware Memory Auto-Scaling

- DAMON kernel module utilizing three DAMOS schemes
- Parameters: min-mem, max-mem, acceptable memory PSI
- Reclaim: Reclaim memory aiming “psi”
- Scale-down: ALLOC/report [min-mem, end) mem aiming “psi”
  - Auto-tune aggressiveness for higher PSI
  - Highest non-fully-DAMOS_ALLOC-ed memory block only
  - Apply ‘struct page’ reduction in some level (like HVO)
- Scale-up: FREE [0, max-mem) mem aiming “psi”
  - Auto-tune aggressiveness for lower PSI
  - Lowest partial-DAMOS_ALLOC-ed memory block only



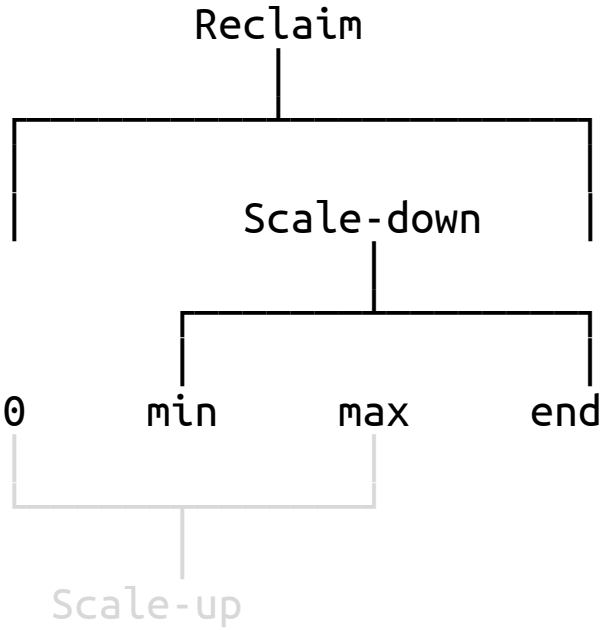
Access/Contiguity-aware Memory Auto-Scaling

- DAMON kernel module utilizing three DAMOS schemes
- Parameters: min-mem, max-mem, acceptable memory PSI
- Reclaim: Reclaim memory aiming “psi”
- Scale-down: ALLOC/report [min-mem, end) mem aiming “psi”
  - Auto-tune aggressiveness for higher PSI
  - Highest non-fully-DAMOS_ALLOC-ed memory block only
  - Apply ‘struct page’ reduction in some level (like HVO)
- Scale-up: FREE [0, max-mem) mem aiming “psi”
  - Auto-tune aggressiveness for lower PSI
  - Lowest partial-DAMOS_ALLOC-ed memory block only



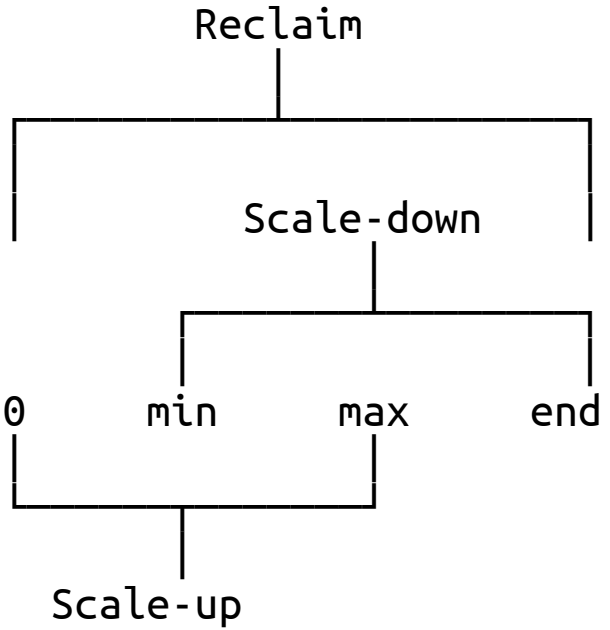
Access/Contiguity-aware Memory Auto-Scaling

- DAMON kernel module utilizing three DAMOS schemes
- Parameters: min-mem, max-mem, acceptable memory PSI
- Reclaim: Reclaim memory aiming “psi”
- Scale-down: ALLOC/report [min-mem, end) mem aiming “psi”
  - Auto-tune aggressiveness for higher PSI
  - Highest non-fully-DAMOS_ALLOC-ed memory block only
  - Apply ‘struct page’ reduction in some level (like HVO)
- Scale-up: FREE [0, max-mem) mem aiming “psi”
  - Auto-tune aggressiveness for lower PSI
  - Lowest partial-DAMOS_ALLOC-ed memory block only



Access/Contiguity-aware Memory Auto-Scaling

- DAMON kernel module utilizing three DAMOS schemes
- Parameters: min-mem, max-mem, acceptable memory PSI
- Reclaim: Reclaim memory aiming “psi”
- Scale-down: ALLOC/report [min-mem, end) mem aiming “psi”
  - Auto-tune aggressiveness for higher PSI
  - Highest non-fully-DAMOS_ALLOC-ed memory block only
  - Apply ‘struct page’ reduction in some level (like HVO)
- Scale-up: FREE [0, max-mem) mem aiming “psi”
  - Auto-tune aggressiveness for lower PSI
  - Lowest partial-DAMOS_ALLOC-ed memory block only



Access-aware Ballooning: Control non-collaborative guests

- Adjust ACMA's max-mem parameter for baloon {in,de}flating
- Reuse virtio-balloon interface (no host-side change)

```
diff --git a/drivers/virtio/virtio_balloon.c b/drivers/virtio/virtio_balloon.c
[...]
@@ -472,6 +472,32 @@ static void virtballoon_changed(struct virtio_device *vdev)
        struct virtio_balloon *vb = vdev->priv;
        unsigned long flags;

+#ifdef CONFIG_ACMA_BALLOON
+       s64 target;
+       u32 num_pages;
+
+       virtio_cread_le(vb->vdev, struct virtio_balloon_config, num_pages,
+                       &num_pages);
+       target = ALIGN(num_pages, VIRTIO_BALLOON_PAGES_PER_PAGE);
+       acma_set_max_mem_aggressive(totalram_pages() - target);
+       return;
+#endif
+
        spin_lock_irqsave(&vb->stop_update_lock, flags);
        if (!vb->stop_update) {
                start_update_balloon_size(vb);
```

Limitations (Hopefully) Mitigated

- Complexity of multiple kernel features orchestration
  - ACMA: single module asking three parameters
- Memory hot-unplugging is slow and easy to fail,
  System-level compaction is wasteful and access oblivious
  - ACMA scales down (isolate/migrate) memory in 2 MiB granularity, colder regions first
- Lack of after-report pages control
  - ACMA returns pages under high memory pressure while keeping maximum contiguity
- Lack of non-collaborative guests control
  - Host can just use virtio-balloon

Limitations (Hopefully) Mitigated

- Complexity of multiple kernel features orchestration
  - ACMA: single module asking three parameters
- Memory hot-unplugging is slow and easy to fail,
  System-level compaction is wasteful and access oblivious
  - ACMA scales down (isolate/migrate) memory in 2 MiB granularity, colder regions first
- Lack of after-report pages control
  - ACMA returns pages under high memory pressure while keeping maximum contiguity
- Lack of non-collaborative guests control
  - Host can just use virtio-balloon

Limitations (Hopefully) Mitigated

- Complexity of multiple kernel features orchestration
  - ACMA: single module asking three parameters
- Memory hot-unplugging is slow and easy to fail,
  System-level compaction is wasteful and access oblivious
  - ACMA scales down (isolate/migrate) memory in 2 MiB granularity, colder regions first
- Lack of after-report pages control
  - ACMA returns pages under high memory pressure while keeping maximum contiguity
- Lack of non-collaborative guests control
  - Host can just use virtio-balloon

Limitations (Hopefully) Mitigated

- Complexity of multiple kernel features orchestration
  - ACMA: single module asking three parameters
- Memory hot-unplugging is slow and easy to fail,
  System-level compaction is wasteful and access oblivious
  - ACMA scales down (isolate/migrate) memory in 2 MiB granularity, colder regions first
- Lack of after-report pages control
  - ACMA returns pages under high memory pressure while keeping maximum contiguity
- Lack of non-collaborative guests control
  - Host can just use virtio-balloon

More Hopeful Usages of Access-aware Contiguous Memory Allocation

- Dynamic contiguous memory allocation pool
  - Finding optimum pool/zone size in static way is challenging
  - Use ACMA ALLOC-ed memory regions as the pool (don’t report to the host)
- DRAM power saving
  - Hot-unplug and power-off fully ACMA ALLOC-ed memory blocks

Progress

- Initial idea was shared on Kernel Summit 2023
- Second version of the design and pseudo-code level implementation example are available

More Future Plans

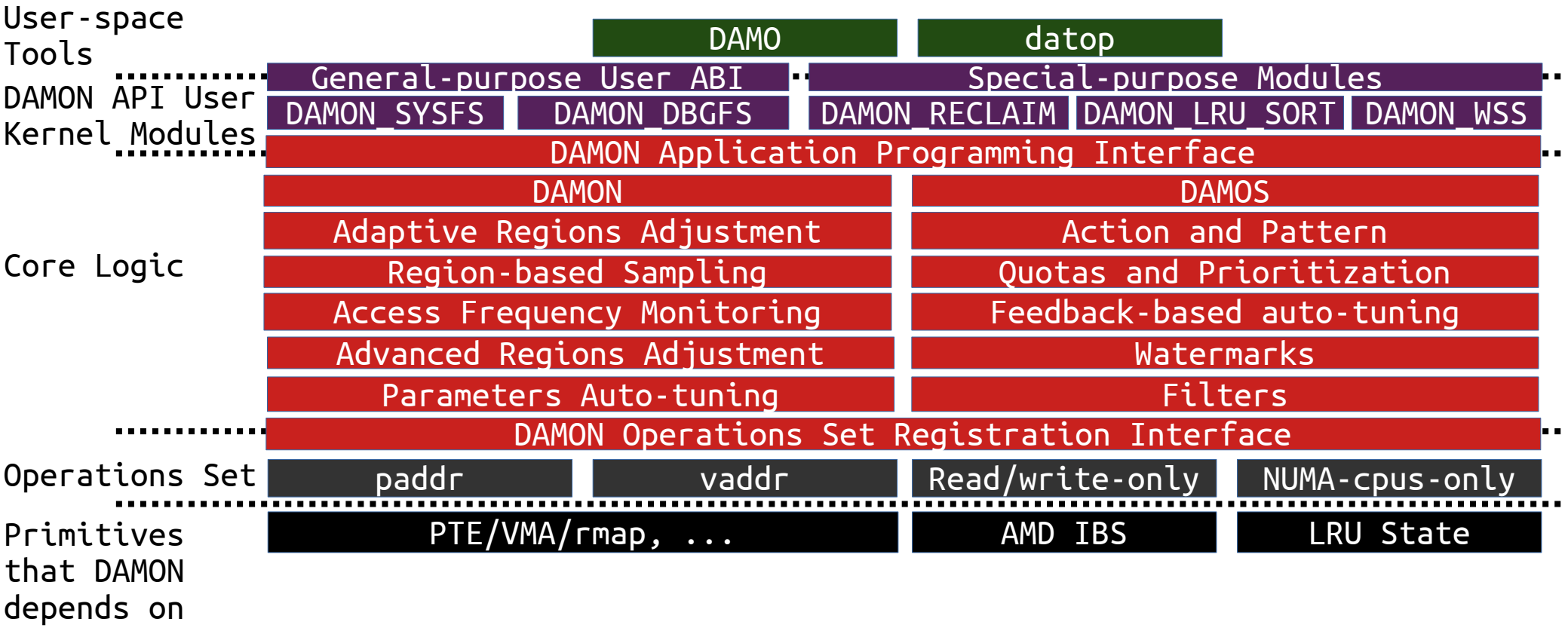
- Monitoring improvements
  - Auto-tuning
  - higher accuracy
- Write-only monitoring
- LRU-sort auto-tuning
- Access-aware THP assistant
- CPU-aware monitoring and NUMA-balancing

Discussion Time!

- ACMA
  - Is there existing alternatives for the motivation use case (memory over-commit VM systems)?
  - Ok to reuse pages reporting from ACMA?
  - Ok to reuse virtio-balloon's interface for Access-aware Ballooning?
  - Will access-aware migration make real improvement? Recommending test workloads?
  - Do DAMOS_ALLOC-based dynamic CMA pool alloc and DRAM power saving make sense?
- Tiered-memory
  - Directly migrate to appropriate tier, instead of incremental bubbling up/down?
  - Any DAMON tuning failures from your tiering approach?
- Questions or comments on updated features and other future plans
- Don't forget sj@kernel.org, damon@lists.linux.dev, and DAMON Beer/Coffee/Tea Chat

Backup Slides

DAMON Stack, In a Future



ACMA and Unmovable/Long-pinned Pages

- Unmovable pages or long-pinned pages can interfere ACMA scale down
  - ACMA apply DAMOS_ALLOC to only not-yet-completely DAMOS_ALLOC-ed memory block of highest address
- Solution: Allow limited amount of not-DAMOS_ALLOC-ed regions in scaling window
- If the ‘struct page’ reduction mechanism can be applied in only memory block granularity (e.g., memory hot-unplugging), ‘struct page’ reduction rate can be reduced
  - Hugetlb vmemmap optimization (HVO)-like approach could be applied instead
  - For DRAM power saving, HVO-like approach cannot help, though