Design

Configurable Layers

DAMON provides data access monitoring functionality while making the accuracy and the overhead controllable. The fundamental access monitorings require primitives that dependent on and optimized for the target address space. On the other hand, the accuracy and overhead tradeoff mechanism, which is the core of DAMON, is in the pure logic space. DAMON separates the two parts in different layers and defines its interface to allow various low level primitives implementations configurable with the core logic. We call the low level primitives implementations monitoring operations.

Due to this separated design and the configurable interface, users can extend DAMON for any address space by configuring the core logics with appropriate monitoring operations. If appropriate one is not provided, users can implement the operations on their own.

For example, physical memory, virtual memory, swap space, those for specific processes, NUMA nodes, files, and backing memory devices would be supportable. Also, if some architectures or devices support special optimized access check primitives, those will be easily configurable.

Reference Implementations of Address Space Specific Monitoring Operations

The monitoring operations are defined in two parts:

1. Identification of the monitoring target address range for the address space.
2. Access check of specific address range in the target space.

DAMON currently provides the implementations of the operations for the physical and virtual address spaces. Below two subsections describe how those work.

VMA-based Target Address Range Construction

This is only for the virtual address space monitoring operations implementation. That for the physical address space simply asks users to manually set the monitoring target address ranges.

Only small parts in the super-huge virtual address space of the processes are mapped to the physical memory and accessed. Thus, tracking the unmapped address regions is just wasteful. However, because DAMON can deal with some level of noise using the adaptive regions adjustment mechanism, tracking every mapping is not strictly required but could even incur a high overhead in some cases. That said, too huge unmapped areas inside the monitoring target should be removed to not take the time for the adaptive mechanism.

For the reason, this implementation converts the complex mappings to three distinct regions that cover every mapped area of the address space. The two gaps between the three regions are the two biggest unmapped areas in the given address space. The two biggest unmapped areas would be the gap between the heap and the uppermost mmap()-ed region, and the gap between the lowermost mmap()-ed region and the stack in most of the cases. Because these gaps are exceptionally huge in usual address spaces, excluding these will be sufficient to make a reasonable trade-off. Below shows this in detail:

```
<heap>
<BIG UNMAPPED REGION 1>
<uppermost mmap()-ed region>
(small mmap()-ed regions and munmap()-ed regions)
<lowermost mmap()-ed region>
<BIG UNMAPPED REGION 2>
<stack>
```

PTE Accessed-bit Based Access Check

Both of the implementations for physical and virtual address spaces use PTE Accessed-bit for basic access checks. Only one difference is the way of finding the relevant PTE Accessed bit(s) from the address. While the implementation for the virtual address walks the page table for the target task of the address, the implementation for the physical address walks every page table having a mapping to the address. In this way, the implementations find and clear the bit(s) for next sampling target address and checks whether the bit(s) set again after one sampling period. This could disturb other kernel subsystems using the Accessed bits, namely Idle page tracking and the reclaim logic. DAMON does nothing to avoid disturbing Idle page tracking, so handling the interference is the responsibility of sysadmins. However, it solves the conflict with the reclaim logic using `PG_idle` and `PG_young` page flags, as Idle page tracking does.

Address Space Independent Core Mechanisms

Below four sections describe each of the DAMON core mechanisms and the five monitoring attributes, `sampling interval`, `aggregation interval`, `update interval`, `minimum number of regions`, and `maximum number of regions`.

Access Frequency Monitoring

The output of DAMON says what pages are how frequently accessed for a given duration. The resolution of the access frequency is controlled by setting `sampling interval` and `aggregation interval`. In detail, DAMON checks access to each page per `sampling interval` and aggregates the results. In other words, counts the number of the accesses to each page. After each `aggregation interval` passes, DAMON calls callback functions that previously registered by users so that users can read the aggregated results and then clears the results. This can be described in below simple pseudo-code:

```
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)
```

The monitoring overhead of this mechanism will arbitrarily increase as the size of the target workload grows.

Region Based Sampling

To avoid the unbounded increase of the overhead, DAMON groups adjacent pages that assumed to have the same access frequencies into a region. As long as the assumption (pages in a region have the same access frequencies) is kept, only one page in the region is required to be checked. Thus, for each `sampling interval`, DAMON randomly picks one page in each region, waits for one `sampling interval`, checks whether the page is accessed meanwhile, and increases the access frequency of the region if so. Therefore, the monitoring overhead is controllable by setting the number of regions. DAMON allows users to set the minimum and the maximum number of regions for the trade-off.

This scheme, however, cannot preserve the quality of the output if the assumption is not guaranteed.

Adaptive Regions Adjustment

Even somehow the initial monitoring target regions are well constructed to fulfill the assumption (pages in same region have similar access frequencies), the data access pattern can be dynamically changed. This will result in low monitoring quality. To keep the assumption as much as possible, DAMON adaptively merges and splits each region based on their access frequency.

For each `aggregation interval`, it compares the access frequencies of adjacent regions and merges those if the frequency difference is small. Then, after it reports and clears the aggregated access frequency of each region, it splits each region into two or three regions if the total number of regions will not exceed the user-specified maximum number of regions after the split.

In this way, DAMON provides its best-effort quality and minimal overhead while keeping the bounds users set for their trade-off.

Dynamic Target Space Updates Handling

The monitoring target address range could dynamically changed. For example, virtual memory could be dynamically mapped and unmapped. Physical memory could be hot-plugged.

As the changes could be quite frequent in some cases, DAMON allows the monitoring operations to check dynamic changes including memory mapping changes and applies it to monitoring operations-related data structures such as the abstracted monitoring target memory area only for each of a user-specified time interval (`update interval`).