Getting Started

This document briefly describes how you can use DAMON by demonstrating its default user space tool. Please note that this document describes only a part of its features for brevity. Please refer to the usage doc of the tool for more details.

Prerequisites

Kernel

You should first ensure your system is running on a kernel built with `CONFIG_DAMON_*=y`.

User Space Tool

For the demonstration, we will use the default user space tool for DAMON, called DAMON Operator (DAMO). It is available at https://github.com/awslabs/damo. The examples below assume that `damo` is on your `$PATH`. It's not mandatory, though.

Because DAMO is using the debugfs interface (refer to :doc:`usage` for the detail) of DAMON, you should ensure debugfs is mounted. Mount it manually as below:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\mm\damon\(linux-master)(Documentation)(admin-guide)(mm)(damon)start.rst, line 32); *backlink***
>
> Unknown interpreted text role "doc".

```
# mount -t debugfs none /sys/kernel/debug/
```

or append the following line to your `/etc/fstab` file so that your system can automatically mount debugfs upon booting:

```
debugfs /sys/kernel/debug debugfs defaults 0 0
```

Recording Data Access Patterns

The commands below record the memory access patterns of a program and save the monitoring results to a file.

```
$ git clone https://github.com/sjp38/masim
$ cd masim; make; ./masim ./configs/zigzag.cfg &
$ sudo damo record -o damon.data $(pidof masim)
```

The first two lines of the commands download an artificial memory access generator program and run it in the background. The generator will repeatedly access two 100 MiB sized memory regions one by one. You can substitute this with your real workload. The last line asks `damo` to record the access pattern in the `damon.data` file.

Visualizing Recorded Patterns

You can visualize the pattern in a heatmap, showing which memory region (x-axis) got accessed when (y-axis) and how frequently (number).:

```
$ sudo damo report heats --heatmap stdout
2222222222222222222222222222222222222211111111111111111111111111111111111100
4444444444444444444444444444444444444343444444444444444444444444444444443200
4444444444444444444444444444444444444433444444444444444444444444444444444200
3333333333333333333333333333333333333334455555555555555555555555555555555200
3333333333333333333333333333333333333344444444444444444444444444444444444200
2222222222222222222222222222222222222335555555555555555555555555555555555200
0000000000000000000000000000000000000028888888888888888888888888888888888400
0000000000000000000000000000000000000028888888888888888888888888888888888400
3333333333333333333333333333333333333335555555555555555555555555555555555200
8888888888888888888888888888888888888860000000000000000000000000000000000000
8888888888888888888888888888888888888860000000000000000000000000000000000000
3333333333333333333333333333333333333344444444444444444444444444444444443200
0000000000000000000000000000000000000028888888888888888888888888888888888400
[...]
# access_frequency:  0  1  2  3  4  5  6  7  8  9
# x-axis: space (139728247021568-139728453431248: 196.848 MiB)
# y-axis: time (15256597248362-15326899978162: 1 m 10.303 s)
# resolution: 80x40 (2.461 MiB and 1.758 s for each character)
```

You can also visualize the distribution of the working set size, sorted by the size.:

```
$ sudo damo report wss --range 0 101 10
# <percentile> <wss>
```

```
# target_id     18446632103789443072
# avr:  107.708 MiB
  0             0 B |                                                           |
 10      95.328 MiB |***************************                                |
 20      95.332 MiB |***************************                                |
 30      95.340 MiB |***************************                                |
 40      95.387 MiB |***************************                                |
 50      95.387 MiB |***************************                                |
 60      95.398 MiB |***************************                                |
 70      95.398 MiB |***************************                                |
 80      95.504 MiB |***************************                                |
 90     190.703 MiB |***********************************************************  |
100     196.875 MiB |************************************************************|
```

Using `--sortby` option with the above command, you can show how the working set size has chronologically changed.:

```
$ sudo damo report wss --range 0 101 10 --sortby time
# <percentile> <wss>
# target_id     18446632103789443072
# avr:  107.708 MiB
  0       3.051 MiB |                                                           |
 10     190.703 MiB |***********************************************************|
 20      95.336 MiB |***************************                                |
 30      95.328 MiB |***************************                                |
 40      95.387 MiB |***************************                                |
 50      95.332 MiB |***************************                                |
 60      95.320 MiB |***************************                                |
 70      95.398 MiB |***************************                                |
 80      95.398 MiB |***************************                                |
 90      95.340 MiB |***************************                                |
100      95.398 MiB |***************************                                |
```

Data Access Pattern Aware Memory Management

Below three commands make every memory region of size >=4K that doesn't accessed for >=60 seconds in your workload to be swapped out.

```
$ echo "#min-size max-size min-acc max-acc min-age max-age action" > test_scheme
$ echo "4K        max      0       0       60s     max     pageout" >> test_scheme
$ damo schemes -c test_scheme <pid of your workload>
```