DM statistics

Device Mapper supports the collection of I/O statistics on user-defined regions of a DM device. If no regions are defined no statistics are collected so there isn't any performance impact. Only bio-based DM devices are currently supported.

Each user-defined region specifies a starting sector, length and step. Individual statistics will be collected for each step-sized area within the range specified.

The I/O statistics counters for each step-sized area of a region are in the same format as */sys/block/*/stat* or */proc/diskstats* (see: Documentation/admin-guide/iostats.rst). But two extra counters (12 and 13) are provided: total time spent reading and writing. When the histogram argument is used, the 14th parameter is reported that represents the histogram of latencies. All these counters may be accessed by sending the @stats_print message to the appropriate DM device via dmsetup.

The reported times are in milliseconds and the granularity depends on the kernel ticks. When the option precise_timestamps is used, the reported times are in nanoseconds.

Each region has a corresponding unique identifier, which we call a region_id, that is assigned when the region is created. The region_id must be supplied when querying statistics about the region, deleting the region, etc. Unique region_ids enable multiple userspace programs to request and process statistics for the same DM device without stepping on each other's data.

The creation of DM statistics will allocate memory via kmalloc or fallback to using vmalloc space. At most, 1/4 of the overall system memory may be allocated by DM statistics. The admin can see how much memory is used by reading:

> /sys/module/dm_mod/parameters/stats_current_allocated_bytes

Messages

@stats_create <range> <step> [<number_of_optional_arguments> <optional_arguments>...] [<program_id> [<aux_data>]]

> Create a new region and return the region_id.
>
> <range>
>
> > "-"
> > > whole device
> >
> > "<start_sector>+<length>"
> > > a range of <length> 512-byte sectors starting with <start_sector>.
>
> <step>
>
> > "<area_size>"
> > > the range is subdivided into areas each containing <area_size> sectors.
> >
> > "/<number_of_areas>"
> > > the range is subdivided into the specified number of areas.
>
> <number_of_optional_arguments>
>
> > The number of optional arguments
>
> <optional_arguments>
>
> > The following optional arguments are supported:
> >
> > precise_timestamps
> > > use precise timer with nanosecond resolution instead of the "jiffies" variable. When this argument is used, the resulting times are in nanoseconds instead of milliseconds. Precise timestamps are a little bit slower to obtain than jiffies-based timestamps.
> >
> > histogram:n1,n2,n3,n4,...
> > > collect histogram of latencies. The numbers n1, n2, etc are times that represent the boundaries of the histogram. If precise_timestamps is not used, the times are in milliseconds, otherwise they are in nanoseconds. For each range, the kernel will report the number of requests that completed within this range. For example, if we use "histogram:10,20,30", the kernel will report four numbers a:b:c:d. a is the number of requests that took 0-10 ms to complete, b is the number of requests that took 10-20 ms to complete, c is the number of requests that took 20-30 ms to complete and d is the number of requests that took more than 30 ms to complete.
>
> <program_id>
>
> > An optional parameter. A name that uniquely identifies the userspace owner of the range. This groups ranges together so that userspace programs can identify the ranges they created and ignore those created by others. The kernel returns this string back in the output of @stats_list message, but it doesn't use it for anything else. If we omit the number of optional arguments, program id must not be a number,

otherwise it would be interpreted as the number of optional arguments.

<aux_data>

An optional parameter. A word that provides auxiliary data that is useful to the client program that created the range. The kernel returns this string back in the output of @stats_list message, but it doesn't use this value for anything.

@stats_delete <region_id>

Delete the region with the specified id.

<region_id>
region_id returned from @stats_create

@stats_clear <region_id>

Clear all the counters except the in-flight i/o counters.

<region_id>
region_id returned from @stats_create

@stats_list [<program_id>]

List all regions registered with @stats_create.

<program_id>
An optional parameter. If this parameter is specified, only matching regions are returned. If it is not specified, all regions are returned.

Output format:

```
<region_id>: <start_sector>+<length> <step> <program_id> <aux_data>
        precise_timestamps histogram:n1,n2,n3,...
```

The strings "precise_timestamps" and "histogram" are printed only if they were specified when creating the region.

@stats_print <region_id> [<starting_line> <number_of_lines>]

Print counters for each step-sized area of a region.

<region_id>
region_id returned from @stats_create

<starting_line>
The index of the starting line in the output. If omitted, all lines are returned.

<number_of_lines>
The number of lines to include in the output. If omitted, all lines are returned.

Output format for each step-sized area of a region:

```
<start_sector>+<length>
        counters
```

The first 11 counters have the same meaning as */sys/block/*/stat or /proc/diskstats*.

Please refer to Documentation/admin-guide/iostats.rst for details.

1. the number of reads completed
2. the number of reads merged
3. the number of sectors read
4. the number of milliseconds spent reading
5. the number of writes completed
6. the number of writes merged
7. the number of sectors written
8. the number of milliseconds spent writing
9. the number of I/Os currently in progress
10. the number of milliseconds spent doing I/Os
11. the weighted number of milliseconds spent doing I/Os

Additional counters:

12. the total time spent reading in milliseconds
13. the total time spent writing in milliseconds

@stats_print_clear <region_id> [<starting_line> <number_of_lines>]

Atomically print and then clear all the counters except the in-flight i/o counters. Useful when the client consuming the statistics does not want to lose any statistics (those updated between printing and clearing).

<region_id>
region_id returned from @stats_create

<starting_line>
: The index of the starting line in the output. If omitted, all lines are printed and then cleared.

<number_of_lines>
: The number of lines to process. If omitted, all lines are printed and then cleared.

@stats_set_aux <region_id> <aux_data>

Store auxiliary data aux_data for the specified region.

<region_id>
: region_id returned from @stats_create

<aux_data>
: The string that identifies data which is useful to the client program that created the range. The kernel returns this string back in the output of @stats_list message, but it doesn't use this value for anything.

Examples

Subdivide the DM device 'vol' into 100 pieces and start collecting statistics on them:

```
dmsetup message vol 0 @stats_create - /100
```

Set the auxiliary data string to "foo bar baz" (the escape for each space must also be escaped, otherwise the shell will consume them):

```
dmsetup message vol 0 @stats_set_aux 0 foo\\ bar\\ baz
```

List the statistics:

```
dmsetup message vol 0 @stats_list
```

Print the statistics:

```
dmsetup message vol 0 @stats_print 0
```

Delete the statistics:

```
dmsetup message vol 0 @stats_delete 0
```