Net DIM - Generic Network Dynamic Interrupt Moderation

Author: Tal Gilboa <talgi@mellanox.com>

Contents

Assumptions

This document assumes the reader has basic knowledge in network drivers and in general interrupt moderation.

Introduction

Dynamic Interrupt Moderation (DIM) (in networking) refers to changing the interrupt moderation configuration of a channel in order to optimize packet processing. The mechanism includes an algorithm which decides if and how to change moderation parameters for a channel, usually by performing an analysis on runtime data sampled from the system. Net DIM is such a mechanism. In each iteration of the algorithm, it analyses a given sample of the data, compares it to the previous sample and if required, it can decide to change some of the interrupt moderation configuration fields. The data sample is composed of data bandwidth, the number of packets and the number of events. The time between samples is also measured. Net DIM compares the current and the previous data and returns an adjusted interrupt moderation configuration object. In some cases, the algorithm might decide not to change anything. The configuration fields are the minimum duration (microseconds) allowed between events and the maximum number of wanted packets per event. The Net DIM algorithm ascribes importance to increase bandwidth over reducing interrupt rate.

Net DIM Algorithm

Each iteration of the Net DIM algorithm follows these steps:

1. Calculates new data sample.
2. Compares it to previous sample.
3. Makes a decision - suggests interrupt moderation configuration fields.
4. Applies a schedule work function, which applies suggested configuration.

The first two steps are straightforward, both the new and the previous data are supplied by the driver registered to Net DIM. The previous data is the new data supplied to the previous iteration. The comparison step checks the difference between the new and previous data and decides on the result of the last step. A step would result as "better" if bandwidth increases and as "worse" if bandwidth reduces. If there is no change in bandwidth, the packet rate is compared in a similar fashion - increase == "better" and decrease == "worse". In case there is no change in the packet rate as well, the interrupt rate is compared. Here the algorithm tries to optimize for lower interrupt rate so an increase in the interrupt rate is considered "worse" and a decrease is considered "better". Step #2 has an optimization for avoiding false results: it only considers a difference between samples as valid if it is greater than a certain percentage. Also, since Net DIM does not measure anything by itself, it assumes the data provided by the driver is valid.

Step #3 decides on the suggested configuration based on the result from step #2 and the internal state of the algorithm. The states reflect the "direction" of the algorithm: is it going left (reducing moderation), right (increasing moderation) or standing still. Another optimization is that if a decision to stay still is made multiple times, the interval between iterations of the algorithm would increase in order to reduce calculation overhead. Also, after "parking" on one of the most left or most right decisions, the algorithm may decide to verify this decision by taking a step in the other direction. This is done in order to avoid getting stuck in a "deep sleep" scenario. Once a decision is made, an interrupt moderation configuration is selected from the predefined profiles.

The last step is to notify the registered driver that it should apply the suggested configuration. This is done by scheduling a work function, defined by the Net DIM API and provided by the registered driver.

As you can see, Net DIM itself does not actively interact with the system. It would have trouble making the correct decisions if the wrong data is supplied to it and it would be useless if the work function would not apply the suggested configuration. This does, however, allow the registered driver some room for manoeuvre as it may provide partial data or ignore the algorithm suggestion under some conditions.

Registering a Network Device to DIM

Net DIM API exposes the main function net_dim(). This function is the entry point to the Net DIM algorithm and has to be called

every time the driver would like to check if it should change interrupt moderation parameters. The driver should provide two data structures: :c:type:`struct dim <dim>` and :c:type:`struct dim_sample <dim_sample>`. :c:type:`struct dim <dim>` describes the state of DIM for a specific object (RX queue, TX queue, other queues, etc.). This includes the current selected profile, previous data samples, the callback function provided by the driver and more. :c:type:`struct dim_sample <dim_sample>` describes a data sample, which will be compared to the data sample stored in :c:type:`struct dim <dim>` in order to decide on the algorithm's next step. The sample should include bytes, packets and interrupts, measured by the driver.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]net_dim.rst, line 88); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]net_dim.rst, line 88); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]net_dim.rst, line 88); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]net_dim.rst, line 88); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]net_dim.rst, line 88); *backlink*

Unknown interpreted text role "c:type".

In order to use Net DIM from a networking driver, the driver needs to call the main net_dim() function. The recommended method is to call net_dim() on each interrupt. Since Net DIM has a built-in moderation and it might decide to skip iterations under certain conditions, there is no need to moderate the net_dim() calls as well. As mentioned above, the driver needs to provide an object of type :c:type:`struct dim <dim>` to the net_dim() function call. It is advised for each entity using Net DIM to hold a :c:type:`struct dim <dim>` as part of its data structure and use it as the main Net DIM API object. The :c:type:`struct dim_sample <dim_sample>` should hold the latest bytes, packets and interrupts count. No need to perform any calculations, just include the raw data.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]net_dim.rst, line 103); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]net_dim.rst, line 103); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]net_dim.rst, line 103); *backlink*

Unknown interpreted text role "c:type".

The net_dim() call itself does not return anything. Instead Net DIM relies on the driver to provide a callback function, which is called when the algorithm decides to make a change in the interrupt moderation parameters. This callback will be scheduled and run in a separate thread in order not to add overhead to the data flow. After the work is done, Net DIM algorithm needs to be set to the

proper state in order to move to the next iteration.

Example

The following code demonstrates how to register a driver to Net DIM. The actual usage is not complete but it should make the outline of the usage clear.

```
#include <linux/dim.h>

/* Callback for net DIM to schedule on a decision to change moderation */
void my_driver_do_dim_work(struct work_struct *work)
{
      /* Get struct dim from struct work_struct */
      struct dim *dim = container_of(work, struct dim,
                                     work);
      /* Do interrupt moderation related stuff */
      ...

      /* Signal net DIM work is done and it should move to next iteration */
      dim->state = DIM_START_MEASURE;
}

/* My driver's interrupt handler */
int my_driver_handle_interrupt(struct my_driver_entity *my_entity, ...)
{
      ...
      /* A struct to hold current measured data */
      struct dim_sample dim_sample;
      ...
      /* Initiate data sample struct with current data */
      dim_update_sample(my_entity->events,
                        my_entity->packets,
                        my_entity->bytes,
                        &dim_sample);
      /* Call net DIM */
      net_dim(&my_entity->dim, dim_sample);
      ...
}

/* My entity's initialization function (my_entity was already allocated) */
int my_driver_init_my_entity(struct my_driver_entity *my_entity, ...)
{
      ...
      /* Initiate struct work_struct with my driver's callback function */
      INIT_WORK(&my_entity->dim.work, my_driver_do_dim_work);
      ...
}
```

Dynamic Interrupt Moderation (DIM) library API

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\[linux-master][Documentation][networking]net_dim.rst, line 175)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/linux/dim.h
    :internal:
```