dm-dust

This target emulates the behavior of bad sectors at arbitrary locations, and the ability to enable the emulation of the failures at an arbitrary time.

This target behaves similarly to a linear target. At a given time, the user can send a message to the target to start failing read requests on specific blocks (to emulate the behavior of a hard disk drive with bad sectors).

When the failure behavior is enabled (i.e.: when the output of "dmsetup status" displays "fail_read_on_bad_block"), reads of blocks in the "bad block list" will fail with EIO ("Input/output error").

Writes of blocks in the "bad block list will result in the following:

1. Remove the block from the "bad block list".
2. Successfully complete the write.

This emulates the "remapped sector" behavior of a drive with bad sectors.

Normally, a drive that is encountering bad sectors will most likely encounter more bad sectors, at an unknown time or location. With dm-dust, the user can use the "addbadblock" and "removebadblock" messages to add arbitrary bad blocks at new locations, and the "enable" and "disable" messages to modulate the state of whether the configured "bad blocks" will be treated as bad, or bypassed. This allows the pre-writing of test data and metadata prior to simulating a "failure" event where bad sectors start to appear.

Table parameters

<device_path> <offset> <blksz>

Mandatory parameters:

<device_path>:

Path to the block device.

<offset>:

Offset to data area from start of device_path

<blksz>:

Block size in bytes

(minimum 512, maximum 1073741824, must be a power of 2)

Usage instructions

First, find the size (in 512-byte sectors) of the device to be used:

```
$ sudo blockdev --getsz /dev/vdb1
33552384
```

Create the dm-dust device: (For a device with a block size of 512 bytes)

```
$ sudo dmsetup create dust1 --table '0 33552384 dust /dev/vdb1 0 512'
```

(For a device with a block size of 4096 bytes)

```
$ sudo dmsetup create dust1 --table '0 33552384 dust /dev/vdb1 0 4096'
```

Check the status of the read behavior ("bypass" indicates that all I/O will be passed through to the underlying device; "verbose" indicates that bad block additions, removals, and remaps will be verbosely logged):

```
$ sudo dmsetup status dust1
0 33552384 dust 252:17 bypass verbose

$ sudo dd if=/dev/mapper/dust1 of=/dev/null bs=512 count=128 iflag=direct
128+0 records in
128+0 records out

$ sudo dd if=/dev/zero of=/dev/mapper/dust1 bs=512 count=128 oflag=direct
128+0 records in
128+0 records out
```

Adding and removing bad blocks

At any time (i.e.: whether the device has the "bad block" emulation enabled or disabled), bad blocks may be added or removed from the device via the "addbadblock" and "removebadblock" messages:

```
$ sudo dmsetup message dust1 0 addbadblock 60
kernel: device-mapper: dust: badblock added at block 60

$ sudo dmsetup message dust1 0 addbadblock 67
kernel: device-mapper: dust: badblock added at block 67

$ sudo dmsetup message dust1 0 addbadblock 72
kernel: device-mapper: dust: badblock added at block 72
```

These bad blocks will be stored in the "bad block list". While the device is in "bypass" mode, reads and writes will succeed:

```
$ sudo dmsetup status dust1
0 33552384 dust 252:17 bypass
```

Enabling block read failures

To enable the "fail read on bad block" behavior, send the "enable" message:

```
$ sudo dmsetup message dust1 0 enable
kernel: device-mapper: dust: enabling read failures on bad sectors

$ sudo dmsetup status dust1
0 33552384 dust 252:17 fail_read_on_bad_block
```

With the device in "fail read on bad block" mode, attempting to read a block will encounter an "Input/output error":

```
$ sudo dd if=/dev/mapper/dust1 of=/dev/null bs=512 count=1 skip=67 iflag=direct
dd: error reading '/dev/mapper/dust1': Input/output error
0+0 records in
0+0 records out
0 bytes copied, 0.00040651 s, 0.0 kB/s
```

...and writing to the bad blocks will remove the blocks from the list, therefore emulating the "remap" behavior of hard disk drives:

```
$ sudo dd if=/dev/zero of=/dev/mapper/dust1 bs=512 count=128 oflag=direct
128+0 records in
128+0 records out

kernel: device-mapper: dust: block 60 removed from badblocklist by write
kernel: device-mapper: dust: block 67 removed from badblocklist by write
kernel: device-mapper: dust: block 72 removed from badblocklist by write
kernel: device-mapper: dust: block 87 removed from badblocklist by write
```

Bad block add/remove error handling

Attempting to add a bad block that already exists in the list will result in an "Invalid argument" error, as well as a helpful message:

```
$ sudo dmsetup message dust1 0 addbadblock 88
device-mapper: message ioctl on dust1  failed: Invalid argument
kernel: device-mapper: dust: block 88 already in badblocklist
```

Attempting to remove a bad block that doesn't exist in the list will result in an "Invalid argument" error, as well as a helpful message:

```
$ sudo dmsetup message dust1 0 removebadblock 87
device-mapper: message ioctl on dust1  failed: Invalid argument
kernel: device-mapper: dust: block 87 not found in badblocklist
```

Counting the number of bad blocks in the bad block list

To count the number of bad blocks configured in the device, run the following message command:

```
$ sudo dmsetup message dust1 0 countbadblocks
```

A message will print with the number of bad blocks currently configured on the device:

```
countbadblocks: 895 badblock(s) found
```

Querying for specific bad blocks

To find out if a specific block is in the bad block list, run the following message command:

```
$ sudo dmsetup message dust1 0 queryblock 72
```

The following message will print if the block is in the list:

```
dust_query_block: block 72 found in badblocklist
```

The following message will print if the block is not in the list:

```
dust_query_block: block 72 not found in badblocklist
```

The "queryblock" message command will work in both the "enabled" and "disabled" modes, allowing the verification of whether a block will be treated as "bad" without having to issue I/O to the device, or having to "enable" the bad block emulation.

Clearing the bad block list

To clear the bad block list (without needing to individually run a "removebadblock" message command for every block), run the following message command:

```
$ sudo dmsetup message dust1 0 clearbadblocks
```

After clearing the bad block list, the following message will appear:

```
dust_clear_badblocks: badblocks cleared
```

If there were no bad blocks to clear, the following message will appear:

```
dust_clear_badblocks: no badblocks found
```

Listing the bad block list

To list all bad blocks in the bad block list (using an example device with blocks 1 and 2 in the bad block list), run the following message command:

```
$ sudo dmsetup message dust1 0 listbadblocks
1
2
```

If there are no bad blocks in the bad block list, the command will execute with no output:

```
$ sudo dmsetup message dust1 0 listbadblocks
```

Message commands list

Below is a list of the messages that can be sent to a dust device:

Operations on blocks (requires a <blknum> argument):

```
addbadblock <blknum>
queryblock <blknum>
removebadblock <blknum>
```

...where <blknum> is a block number within range of the device (corresponding to the block size of the device.)

Single argument message commands:

```
countbadblocks
clearbadblocks
listbadblocks
disable
enable
quiet
```

Device removal

When finished, remove the device via the "dmsetup remove" command:

```
$ sudo dmsetup remove dust1
```

Quiet mode

On test runs with many bad blocks, it may be desirable to avoid excessive logging (from bad blocks added, removed, or "remapped"). This can be done by enabling "quiet mode" via the following message:

```
$ sudo dmsetup message dust1 0 quiet
```

This will suppress log messages from add / remove / removed by write operations. Log messages from "countbadblocks" or "queryblock" message commands will still print in quiet mode.

The status of quiet mode can be seen by running "dmsetup status":

```
$ sudo dmsetup status dust1
0 33552384 dust 252:17 fail_read_on_bad_block quiet
```

To disable quiet mode, send the "quiet" message again:

```
$ sudo dmsetup message dust1 0 quiet

$ sudo dmsetup status dust1
0 33552384 dust 252:17 fail_read_on_bad_block verbose
```

(The presence of "verbose" indicates normal logging.)

"Why not...?"

scsi_debug has a "medium error" mode that can fail reads on one specified sector (sector 0x1234, hardcoded in the source code), but it uses RAM for the persistent storage, which drastically decreases the potential device size.

dm-flakey fails all I/O from all block locations at a specified time frequency, and not a given point in time.

When a bad sector occurs on a hard disk drive, reads to that sector are failed by the device, usually resulting in an error code of EIO ("I/O error") or ENODATA ("No data available"). However, a write to the sector may succeed, and result in the sector becoming readable after the device controller no longer experiences errors reading the sector (or after a reallocation of the sector). However, there may be bad sectors that occur on the device in the future, in a different, unpredictable location.

This target seeks to provide a device that can exhibit the behavior of a bad sector at a known sector location, at a known time, based on a large storage device (at least tens of gigabytes, not occupying system memory).