dt Data Corruption Retries

1. Overview

dt has a default set of data corruption retries, which are shown in the help text:

Retry Related Options:

retry_error=value - The error code to retry.
retry_delay=value - The retry delay. (Def: 5 secs)
retry_limit=value - The retry limit. (Def: 60)
retryDC_delay=value - The retry corruptions delay. (Def: **5**)
retryDC_limit=value - The retry corruptions limit. (Def: **1**)

As you can see, there are various retry options, but those related to data corruption are **DC** options.

The DC options are used during re-reads of the corrupted data.

The data corruption delay is in seconds, and is multiplied by the retry count so 5, 10, 15, ... so trace buffers don't overflow. 🙂

Please know, there's also an **enable=looponerror** which will loop indefinitely, or until *dt* is stopped. Now that dt emits re-read command lines, this flag is not generally required nowadays.

When these options do not match your requirements, simply override the defaults by adding the above options on the dt command line, usually after the workload specified.

2. Data Corruption Re-reads

These are the steps you'll observe in the dt logs:

- The expected and corruption data is saved to files. (controlled via **savecorrupted** flag)
- After the data corruption is detected, the 1st re-read is immediate.

- The re-read data is compared with the expected and previous read.
- Depending on the results of the comparison, dt reports a possible read or write error.
- The delay if non-zero, is then applied (sleeping for N seconds, 5 seconds by default).
- The next re-read is then performed and repeated up until the retry DC limit.

Please know, the delays between re-reads is done to help determine if we're encountering a *permanent* or *transient* corruption.

Permanent corruption means the re-reads are consistently incorrect, so we assume it was written incorrectly.

A *transient* corruption means a subsequent re-read finds the expected data, and usually points to a caching issue.

3. Forcing Data Corruptions

The latest dt has options to create **false** corruption, added to verify triggers and retry DC options:

Force Corruption Options:

corrupt_index=value - The corruption index. (Default: **random**)
corrupt_length=value - The corruption length. (Default: **4 bytes**)
corrupt_pattern=value - The corruption pattern. (Default: **0xfeedface**)
corrupt_step=value - Corruption buffer step. (Default: 0 bytes)
corrupt_reads=value - Corrupt at read records. (Default: **13**)
corrupt_writes=value - Corrupt at write records. (Default: 0)

While the default is to inject **false** read corruptions, the last option will inject write corruptions.

The corruptions will occur at random offsets within the **13th record**, by injecting **0xfeedface**

Note: The length and step options can be used to corrupt more than one block. 🙂

3.1. Corruption Examples

3.1.1. Read File Corruption

This example sets the data corruption (DC) delay to 1 second, and the limit to 10 retries.

Additional options added for this test include:

- **disable=savecorrupted** - disable saving the expected and received data.
- **disable=trigdefaults** - disable the default trigger options (if previously defined).

 # dt if=/mnt/localhost/<path>/dt.data disable=savecorrupted disable=stats,trigdefaults retryDC_delay=1 retryDC_limit=10 workload=log_timestamps

...

2021-05-01,10:32:27 00m00.01s dt (j:1 t:1): Rereading and verifying record data using Direct I/O...
2021-05-01,10:32:27 00m00.01s dt (j:1 t:1): Record #1 - Reading 512 bytes (1 block) into buffer 0x7f7b54003000, Physical LBA 272384 (offset 0)
2021-05-01,10:32:27 00m00.01s dt (j:1 t:1): **Reread data matches previous data read, possible write failure!**
2021-05-01,10:32:27 00m00.01s dt (j:1 t:1): Delaying 1 seconds after retry 1...
2021-05-01,10:32:28 00m01.01s dt (j:1 t:1): Record #1 - Reading 512 bytes (1 block) into buffer 0x7f7b54003000, Physical LBA 272384 (offset 0)
2021-05-01,10:32:28 00m01.01s dt (j:1 t:1): **Reread data matches previous data read, possible write failure!**
2021-05-01,10:32:28 00m01.01s dt (j:1 t:1): Delaying 2 seconds after retry 2...
2021-05-01,10:32:30 00m03.01s dt (j:1 t:1): Record #1 - Reading 512 bytes (1 block) into buffer 0x7f7b54003000, Physical LBA 272384 (offset 0)

...

2021-05-01,10:33:12 00m45.06s dt (j:1 t:1): Command line to re-read the corrupted data:
2021-05-01,10:33:12 00m45.06s dt (j:1 t:1): -> **dt if=/mnt/localhost/<path>/dt.data bs=512 count=1 offset=0 pattern=0x39c39c39 disable=retryDC,savecorrupted,trigdefaults**
2021-05-01,10:33:12 00m45.06s dt (j:1 t:1):
2021-05-01,10:33:12 00m45.06s dt (j:1 t:1): Command line to re-read the corrupted file:
2021-05-01,10:33:12 00m45.06s dt (j:1 t:1): -> **dt if=/mnt/localhost/<path>/dt.data bs=512 dsize=512 iotype=sequential iodir=forward limit=512 records=1 pattern=0x39c39c39 disable=retryDC,savecorrupted,trigdefaults**
2021-05-01,10:33:12 00m45.06s dt (j:1 t:1):

The dt command lines can be used to re-read just the corrupted record or re-read the entire file.

3.1.2. Forced Corruption

root@rtp-smc-qa18-4 /usr/local# **dt of=/mnt/localhost/<path> workload=longevity_file_system limit=25m threads=1 disable=trigdefaults enable=force-corruption retryDC_limit=2**

1 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1):
2 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): SCSI Information:
3 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): **SCSI Device Name: /dev/mapper/mpathbc**
4 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): Vendor Identification: <VID>
5 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): Product Identification: <PID>
6 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): Firmware Revision Level: 1.0
7 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): Target Port Group Support: 1 (implicit ALUA)
8 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): Block Length: 512
9 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): Maximum Capacity: 10737418240 (5242880.000 Mbytes, 5120.000 Gbytes)

10 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): Provisioning Management: Thin Provisioned
11 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): Device Identifier: 7f13-9950-e1e6-0fde-6c9c-e900-7dd0-1493
12 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): Device Serial Number: 7f139950e1e60fde6c9ce9007dd01493
13 2021-05-06,15:53:37 00m00.01s dt (j:1 t:1): Management Network Address: <IP address>

...
102 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): **Warning: Record #13, Corrupting buffer 0x7f6d54bef000 at index 151770, length 4, pattern 0xfeedface, step 0...**
...
140 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): Error Number: 1
141 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): Time of Current Error: Thu May 6 15:53:37 2021
142 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): Write Pass Start Time: Thu May 6 15:53:37 2021
143 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): Pass Number: 3
144 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): Pass Elapsed Time: 00m00.07s
145 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): Test Elapsed Time: 00m00.25s
146 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): File Name: /mnt/localhost/<path>/<file name>.data
147 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): File Inode: 12 (0xc)
148 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): Directory Inode: 2 (0x2)
149 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): File Size: 6569984 (0x644000)
150 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): Operation: **miscompare**
151 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): **Record Number: 13**

168 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): **Starting Physical Error LBA: 299215 (0x490cf)**
169 2021-05-06,15:53:37 00m00.25s dt (j:1 t:1): **Physical Error LBA Offset: 153198080 (0x9219e00)**

...

528 2021-05-06,15:53:37 00m00.26s dt (j:1 t:1):
529 2021-05-06,15:53:37 00m00.27s dt (j:1 t:1): Writing expected data to file <file name>-EXPECT0-j1t1, from buffer 0x7f6d54cf1000, 324096 bytes...
530 2021-05-06,15:53:37 00m00.27s dt (j:1 t:1): Writing corrupted data to file <file name>-CORRUPT0-j1t1, from buffer 0x7f6d54bef000, 324096 bytes...
531 2021-05-06,15:53:37 00m00.27s dt (j:1 t:1): Rereading and verifying record data using Direct I/O...
532 2021-05-06,15:53:37 00m00.27s dt (j:1 t:1): Record #1 - Reading 324096 bytes (633 blocks) into buffer 0x7f6d55aab000, Physical LBA's 298919, 299551 (offset 6245888)
533 2021-05-06,15:53:37 00m00.27s dt (j:1 t:1): **Reread data matches the expected data, possible read failure!**
532 2021-05-06,15:53:37 00m00.27s dt (j:1 t:1): Writing reread data to file <file name>-

REREAD0-j1t1, from buffer 0x7f6d55aab000, 324096 bytes...
534 2021-05-06,15:53:37 00m00.27s dt (j:1 t:1): **Delaying 5 seconds after retry 1...**
535 2021-05-06,15:53:42 00m05.28s dt (j:1 t:1): Record #1 - Reading 324096 bytes (633 blocks) into buffer 0x7f6d55aab000, Physical LBA's 298919, 299551 (offset 6245888)
536 2021-05-06,15:53:42 00m05.28s dt (j:1 t:1): **Reread data matches the expected data, possible read failure!**
533 2021-05-06,15:53:42 00m05.28s dt (j:1 t:1): Writing reread data to file <file name>-REREAD1-j1t1, from buffer 0x7f6d55aab000, 324096 bytes...
534 2021-05-06,15:53:42 00m05.28s dt (j:1 t:1): Command line to re-read the corrupted data:
535 2021-05-06,15:53:42 00m05.28s dt (j:1 t:1): -> **dt if=/mnt/localhost/<path>/<file name>.data bs=324096 count=1 offset=6245888 prefix="/mnt/localhost/<path>/<file name>@<host name>" pattern=iot iotseed=0x03030303 flags=direct enable=btags disable=retryDC,savecorrupted,trigdefaults**

3.1.3. Reference Trace

Today, dt provides file system offset mapping to physical LBA's for Linux and Windows NTFS. For Linux, this mapping has been verified for ext4 and XFS file systems and is accomplished via a file system IOCTL someone thankfully added on Linux. For other OS's and other file systems, dt does **not** provide file system offset to physical LBA mapping, and since most folks test with Linux and Windows, no effort has been to made to add *metadata prowlers* for other OS's. Also know, this mapping does *not* work with logical volumes (LVM's). For LVM's there may be system utilities to provide this mapping, but reference traces can be used instead.

So I've been asked several times, "*How do I acquire this physical LBA mapping on other OS's?*" For example, on Solaris, where I may be using UFS or ZFS file systems, how can I find the physical LBA's of the corrupted record dt reported?

This is the reason why dt reports the re-read command line, as shown above, and is what I'm referring to as a *reference trace!* By enabling host tracing (*tcpdump* or *Wireshark*), an analyzer trace, or array tracing, all you need to do is execute the re-read command line and observe the physical LBA's in the trace.

Also know, while the re-read command line is for the entire record, you can also modify this to re-read just the corrupted blocks or individual blocks, using previous reported corruption information. And in case it's not obvious, please know this *must* be done with the original file system, **not** after copying the file elsewhere! 🙂

3.1.4. Re-reading Corruptions via spt

Using the disk name and the lba/offset reported, tools such as *spt* can be used to re-read the corrupted block(s), and/or loop via **repeat=value** option.

This is esp. useful when the relative file offset mapping is **not** available. This is referred to as a *reference trace*, so the physical LBA is evident in analyzer or IO traces! 🙂

```
root@rtp-smc-qa18-4 ~# spt dsf=/dev/mapper/mpathbc read16 starting=299215 length=512 limit=512 repeat=1k emit=default
spt (j:1 t:1):
spt (j:1 t:1):                        Thread: 1
spt (j:1 t:1):                   Device Name: /dev/mapper/mpathbc
spt (j:1 t:1):                   Device Info: Block Length=512, Capacity=10737418240
spt (j:1 t:1):                   Block Range: 299215 - 299216
spt (j:1 t:1):                     SCSI Name: Read(16)
spt (j:1 t:1):                      SCSI CDB: 88 00 00 00 00 00 00 04 90 cf 00 00 00 01 00 00
spt (j:1 t:1):                Data Direction: read
spt (j:1 t:1):                   Data Length: 512
spt (j:1 t:1):                   Exit Status: 0 = SUCCESS
spt (j:1 t:1):                   Host Status: 0 = DID_OK
spt (j:1 t:1):                 Driver Status: 0 = DRIVER_OK
spt (j:1 t:1):                   SCSI Status: 0 = SCSI_GOOD
spt (j:1 t:1):                    Sense Code: 0 = NO CODE
spt (j:1 t:1):                     Sense Key: 0 = NO SENSE
spt (j:1 t:1):                       asc/ascq: (00, 00) = No additional sense information
spt (j:1 t:1):              Bytes Transferred: 512 (data bytes transferred)
spt (j:1 t:1):                       Residual: 0 (bytes not transferred)
spt (j:1 t:1):                     Iterations: 1024
spt (j:1 t:1):                    Total Bytes: 524288
spt (j:1 t:1):                   Total Blocks: 1024
spt (j:1 t:1):                Total Operations: 1025
spt (j:1 t:1):                     Sense Data: 00 00 00 00 00 00 00 00
spt (j:1 t:1):                   Elapsed Time: 00m00.36s
spt (j:1 t:1):                  Starting Time: Thu May  6 16:13:51 2021
spt (j:1 t:1):                    Ending Time: Thu May  6 16:13:51 2021
spt (j:1 t:1):
root@rtp-smc-qa18-4 ~#
```