

## Contents

- 1 Details of dt's No Progress Feature
  - ◆ 1.1 Long No Progress Times
  - ◆ 1.2 Troubleshooting?
    - ◇ 1.2.1 History Mechanism
    - ◇ 1.2.2 Keepalive Feature
    - ◇ 1.2.3 Trigger Script
  - ◆ 1.3 Unix Implementation
  - ◆ 1.4 Windows Implementation
  - ◆ 1.5 Operation Timing?
  - ◆ 1.6 Examples

## Details of dt's No Progress Feature

This feature was originally added for Unix system to time read/write requests. Later it was extended to time open, close, fsync, and AIO requests. Most recently, this feature was also added to the native Windows *dt*.

The options added to support this feature are:

- `noprogt=value` - The time in seconds when to report no progress.
- `noprogtt= value` - The time in second when to execute a trigger script.
- `alarm=value` - The time in seconds when to check for no progress.
- `trigger=cmd:script` - The (optional) trigger script to execute.
- `enable=tdebug` - Enable timer debug (added to recent versions).
- `notime=otype` - Disable timing of specified operation type:
  - ◆ Valid operation types are:
    - ◇ open close read write ioctl fsync msync aiowait

The timing is around each system call. In the case of AIO, the `aiowait()` API is timed, which is invoked after queuing up the requested AIO's.

Although this feature was added specifically to time I/O's during Cluster Failover (CFO) events, other host side events may cause long times. CFO == SFO (Storage Failover in ONTAP-8).

## Long No Progress Times

Although designed to time each system call, other conditions may cause long times such as:

- file system block allocation (see [BURT 312867](#))
- buffer cache flushing
- resource shortages
- network congestion
- wire data saturation
- excessive paging/swapping
- excessive SCSI Queue Full conditions
- processes hogging the CPUs
- process scheduling (too many procs)

In each case above, the actual I/O may be progressing fine (albeit slowly), but *dt* will be reporting long no-progress times (which may simply indicate slow progress).

Of course, since ONTAP is also an operating system (OS), many of the items listed above also apply to ONTAP, and not just the host (or client) side OS.

Now when long no-progress times occur unexpectedly (no CFO's), that's when things get interesting and oftentimes you'll need to get creative to find root cause: ONTAP or Host OS?, and which particular condition you are hitting?

## Troubleshooting?

As mentioned above, when unexpected no-progress times occur you oftentimes need to get creative and usually need to utilize additional tools, such as *iostat*, *netstat*, *vmstat*, reviewing error log entries or system log files, as well as similar ONTAP tools and logs.

## History Mechanism

*dt* has a timer debug option (`enable=tdebug`), and a new history mechanism that can be used to time every read/write request with its' own timing. The history options are:

- `history=value` - The number of history entries to save (circular buffer)
- `hdsiz=value` - The history data size (defaults to 36 bytes, can be set to 0)
- `enable=htiming` - Enables timing each history entry (uses `gettimeofday()` API)
- `enable=dump` - Enables dumping the history entries at the end of a run

Note: History entries are automatically dumped when an I/O or data corruption occurs.

You'll want to use system utilities to ensure OS conditions mentioned above, are not impacting and creating the long no-progress times.

## Keepalive Feature

Besides the timer debug, one can enable what's known as "keepalive" messages. This feature was originally added for periodic reporting of user defined information while a test is running.

For example:

- `keepalive='Pass: Reads %r, Writes %w Total: Reads %R Writes %W'`

Will report the reads and writes per pass and total (when using runtime or passes options).

For more information of what can be reported, see "dt help" under "Keepalive Format Control:".

## Trigger Script

The trigger script existed prior to adding the no-progress feature. It was added initially so when a failure occurred, such as an I/O error or data corruption, an external script can be invoked to gather additional information and/or trigger an analyzer or panic ONTAP or the host OS to freeze the condition and gather a crash dump (aka "core" here at NetApp).

This script feature was extended for the no-progress feature. When the no-progress trigger time (`noprogtt=value`) is reached and `trigger:cmdscript` was specified, then the external script is invoked with the following arguments:

```
roanoke% more /x/eng/localtest/noarch/bin/dt_noprogram_script.ksh
#!/bin/ksh -p
#
# Script Owner - Thomas Masterson (thomasm@netapp.com) x5778
# Location - /x/eng/localtest/noarch/bin
# We get called with these parameters
# dname op dsize offset position lba errno noprogtt
# Where:
#   $0 Name of the program
#   $1 dname = The device/file name.
#   $2 op = open/close/read/write/miscompare/noprogram
#   $3 dsize = The device block size.
#   $4 offset = The current file offset.
#   $5 position = The failing offset within block.
#   $6 lba = The logical block address (relative for FS).
#   $7 errno = The error number on syscall errors.
#   $8 noprogtt = The no progress time
#   $9 UserScript - User supplied script to also run
#
...

```

So, when the operation "`op == noprogram`", this indicates a too long no-progress time.

The exit status of the script controls what action *dt* takes next:

- `CONTINUE = 0` (tells *dt* to continue running)
- `TERMINATE = 1` (tells *dt* terminate)
- `SLEEP = 2` (*dt* will sleep indefinitely)
- `ABORT = 3` (calls `abort()` API to generate a *dt* core dump)

Note: This exit status only affects no-progress, none of the other operations.

## Unix Implementation

The Unix implementation implemented monitoring via the `alarm()` API, which enables a timer which delivers an alarm signal (`SIGALRM`) when the timer expires. While this works well on some OS's such as AIX, other OS's

may block signals while I/O is in progress, and the signal only gets delivered when returning from the system call. When this occurs, you'll only see one long time reported, rather than noprog= interval messages, and worse of all you'll see nothing when I/O is hung! (sorry)

Note: This signal method will be switched to thread monitoring in a future release.

## Windows Implementation

The native Windows *dt* uses a separate thread for monitoring operations. This is not only more reliable than signals, it was a necessity since alarm() and SIGALRM are not available on Windows.

## Operation Timing?

For those curious about the actual timing code, here's a snippet for read operations:

```

    if (noprog_flag && optiming_table[READ_OP].opt_timing_flag) {
        dip->di_optype = READ_OP;
        dip->di_initiated_time = time((time_t *)0);
    }
#ifdef WIN32
    if (!ReadFile (dip->di_fd, buffer, bsize, &count, NULL)) { count = -1; }
#else /* !defined(WIN32) */
    count = read (dip->di_fd, buffer, bsize);
#endif /* defined(WIN32) */
    if (noprog_flag) {
        dip->di_optype = NONE_OP;
        dip->di_initiated_time = (time_t) 0;
    }

```

The monitoring function also uses time(), who's resolution is seconds, and does the necessary math and compares that with noprog=value to determine if the time you specified is exceeded, then reports this fact.

## Examples

Assuming the following options are used:

- noprog=10s noproggt=300s alarm=5s trigger="cmd:/x/eng/localtest/noarch/bin/dt\_noprog\_script.ksh"

You'll see messages such as:

```

dt.latest (753872): No progress made during aiowait() on /mnt/GPFS_FS_iSCSI3/ibmbc-aix02b3/nate_test
dt.latest (753872): No progress made during aiowait() on /mnt/GPFS_FS_iSCSI3/ibmbc-aix02b3/nate_test
...
dt.latest (753872): No progress made during aiowait() on /mnt/GPFS_FS_iSCSI3/ibmbc-aix02b3/nate_test
...
(753872): No progress made during aiowait() on /mnt/GPFS_FS_iSCSI3/ibmbc-aix02b3/nate_test-753872 fo
(753872): Executing: /x/eng/localtest/noarch/bin/dt_noprog_script.ksh /mnt/GPFS_FS_iSCSI3/ibmbc-aix0
#####
dt trigger scripting being called - /x/eng/localtest/noarch/bin/dt_noprog_script.ksh
#####
The device name is /mnt/GPFS_FS_iSCSI3/ibmbc-aix02b3/nate_test-753872
The operation is noprog

```

## User:Rtmiller/datatest\_(dt)\_No\_Progress\_Details

```
The device block size is 512
The offset, position and lba are 0, 0, 0
The errno is 0
The no progress time is 305
#####
#####
TimeStamp: Wednesday 10Sep2008 04:14:10 PM
#####
#####
/x/eng/localtest/noarch/bin/dt_noprog_script.ksh: I/O has exceeded the limit (noprog)
#####
#####
/x/eng/localtest/noarch/bin/dt_noprog_script.ksh: No additional user supplied script was specified
#####
#####
The return code that /x/eng/localtest/noarch/bin/dt_noprog_script.ksh is sending to dt is 3
#####
dt.latest (753872): Trigger exited with status 3!
dt.latest (753872): Aborting...

Total Statistics (753872):
  Output device/file name: /mnt/GPFS_FS_iSCSI3/ibmbc-aix02b3/nate_test-753872 (device type=regula
  Type of I/O's performed: random (rseed=0x2424645)
  Random I/O Parameters: position=0, ralign=512, rlimit=1073741824
  Current Process Reported: 1/4
  Data pattern string used: 'IOT Pattern' (blocking is 512 bytes)
  Total records processed: 0 with min=2048, max=1048576, incr=variable
  Total bytes transferred: 0 (0.000 Kbytes, 0.000 Mbytes)
  Average transfer rates: 0 bytes/sec, 0.000 Kbytes/sec
  Asynchronous I/O's used: 16
  Number I/O's per second: 0.000
  Total passes completed: 0
  Total errors detected: 0/1
    Total elapsed time: 05m05.36s
    Total system time: 00m00.03s
    Total user time: 00m00.03s
      Starting time: Wed Sep 10 16:09:01 2008
      Ending time: Wed Sep 10 16:14:10 2008

dt.latest: Child process 753872, exiting because of signal 6
```

This is a new problem, which has not been root caused yet, but it's apparent from the statistics that no writes have completed within 5 minutes.