# 1 Overview

This release contains several bug fixes and enhancements, including:

- Fixes issue with variable file sizes and random percentages.
- Collects and reports I/O latency information.
- New predefined workloads.

# 2 Latency Information

The normal I/O paths have been updated to acquire read and write latency in microseconds. The read, write, and total latency is reported in the job or total statistics, but are not reported in the per pass statistics. Options exist to report latency via the keepalive messages, with predefined latency keepalive templates available.

The minimum, maximum, and total latency is collected.

The latency values are scaled to seconds, milliseconds, or microseconds.

Please Note: The Async I/O paths have *not* been updated.

## 2.1 Latency Options

The following options have been added to control latency information.

### 2.1.1 Latency Frequency

This option controls how often to report read and write latency:

```
        latfreq=value           Report latency frequency (in records).
                                Note: This reports the per I/O latency.
```

A *value* of one (1) will report the latency of every I/O.

### 2.1.2 Minimum Latency

This option defines the minimum latency, in microseconds. When latency is less than this value, then I/O is reported as "(fast)".

```
        latmin=value            The minimum latency (in microseconds).
```

### 2.1.3 Maximum Latency

This option defines the maximum latency, in microseconds. When latency exceeds this value then I/O is reported as"(slow)".

```
        latmax=value            The maximum latency (in microseconds).
```

## 2.2 Keepalive Format Control Strings

Keepalive strings are a user defined set of text and format control strings reported by the monitoring thread whenever the keepalive time is reached.

The following keepalive format control strings have been added:

Latency Keywords:

%latency = The average total latency.

%latmin = The min total latency.  %latmax = The max total latency.

%rlatency = The average read latency.

%rlatmin = The min read latency.  %rlatmax = The max read latency.

%wlatency = The average write latency.

%wlatmin = The min write latency. %wlatmax = The max write latency.

## 2.3 Keepalive Workload Templates

Workload templates are used in conjunction with other workloads, and are used to define a set of options useful to most folks and to save them from specifying long command lines. Standard and verbose keepalive, pass, and total keepalive templates were added.

The latency workload templates are:

**latency_keepalive**: Latency Brief Keepalive Message (template)

keepalive='avg/min/max Total Latency: %latency/%latmin/%latmax, Read Latency: %rlatency/%rlatmin/%rlatmax, Write Latency: %wlatency/%wlatmin/%wlatmax'

**platency_keepalive**: Pass Latency Keepalive Message (template)

disable=pstats pkeepalive='Pass %p, avg/min/max Total Latency: %latency/%latmin/%latmax, Read Latency: %rlatency/%rlatmin/%rlatmax, Write Latency: %wlatency/%wlatmin/%wlatmax'

**tlatency_keepalive**: Total Latency Keepalive (template)

workload=platency_keepalive stats=brief tkeepalive='\n    Latency Statistics:\n\tTotal Latency: Average: %latency, Minimum: %latmin, Maximum: %latmax\n\t Read Latency: Average: %rlatency, Minimum: %rlatmin, Maximum: %rlatmax\n\tWrite Latency: Average: %wlatency, Minimum: %wlatmin, Maximum: %wlatmax'

**vlatency_keepalive**: Latency Verbose Keepalive (template)

keepalive='\n    Latency Statistics:\n\tTotal Latency: Average: %latency, Minimum: %latmin, Maximum: %latmax\n\t Read Latency: Average: %rlatency, Minimum: %rlatmin, Maximum: %rlatmax\n\tWrite Latency: Average: %wlatency, Minimum: %wlatmin, Maximum: %wlatmax'

## 3 New Workload

A workload was added to create many directories, rather than many files. Actually, a single file in each directory must be created with the current logic employed in dt.

**many_dirs**: Create many directories (~100k dirs)

bs=1 records=1 sdirs=1000 depth=10 threads=10 dispose=keep

Notes:

- Ten threads consumed 100% of the CPU on my system, so adjust as required.
- On Windows the default path length is 260 characters, thus the depth above being set to 10, since 100 exceeded the path length. The LongPathsEnabled file system registry setting can be enabled if longer paths are desired:

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem]

"**LongPathsEnabled**"=dword:00000001

## 4 Many Directories Example

As most know, I like to teach by example, so these examples will help guide you: 😊

```
$ ./dt of=f workload=many_dirs keepalivet=1m workload=latency_keepalive sdirs=1000 depth=10 threads=10
dt.exe (j:1 t:1): avg/min/max Total Latency: 822us/33us/144.86ms, Read Latency: na/na/na, Write Latency: 822us/33us/144.86ms
dt.exe (j:1 t:2): avg/min/max Total Latency: 815us/34us/58.73ms, Read Latency: na/na/na, Write Latency: 815us/34us/58.73ms
dt.exe (j:1 t:3): avg/min/max Total Latency: 727us/33us/20.37ms, Read Latency: na/na/na, Write Latency: 727us/33us/20.37ms
dt.exe (j:1 t:4): avg/min/max Total Latency: 781us/31us/37.88ms, Read Latency: na/na/na, Write Latency: 781us/31us/37.88ms
dt.exe (j:1 t:5): avg/min/max Total Latency: 815us/35us/16.92ms, Read Latency: na/na/na, Write Latency: 815us/35us/16.92ms
dt.exe (j:1 t:6): avg/min/max Total Latency: 791us/32us/19.93ms, Read Latency: na/na/na, Write Latency: 791us/32us/19.93ms
dt.exe (j:1 t:7): avg/min/max Total Latency: 829us/34us/148.49ms, Read Latency: na/na/na, Write Latency: 829us/34us/148.49ms
dt.exe (j:1 t:8): avg/min/max Total Latency: 815us/32us/85.63ms, Read Latency: na/na/na, Write Latency: 815us/32us/85.63ms
dt.exe (j:1 t:9): avg/min/max Total Latency: 902us/35us/173.99ms, Read Latency: na/na/na, Write Latency: 902us/35us/173.99ms
dt.exe (j:1 t:10): avg/min/max Total Latency: 868us/34us/143.33ms, Read Latency: na/na/na, Write Latency: 868us/34us/143.33ms
        …
dt.exe (j:1 t:9): Total Statistics:
dt.exe (j:1 t:9):        Output device/file name: d0\j1t9\d1000\d2\d3\d4\d5\d6\d7\d8\d9\d10\f
(device type=regular)
dt.exe (j:1 t:9):        Type of I/O's performed: sequential (forward)
dt.exe (j:1 t:9):       Job Information Reported: Job 1, Thread 9
dt.exe (j:1 t:9):        Current Thread Reported: 9/10
dt.exe (j:1 t:9):      Data pattern read/written: 0x26673333
dt.exe (j:1 t:9):             Total records read: 10001
dt.exe (j:1 t:9):               Total bytes read: 10001 (9.767 Kbytes, 0.010 Mbytes, 0.000
Gbytes)
dt.exe (j:1 t:9):          Total records written: 10001
dt.exe (j:1 t:9):            Total bytes written: 10001 (9.767 Kbytes, 0.010 Mbytes, 0.000
Gbytes)
```

```
dt.exe (j:1 t:9):        Total records processed: 20002 @ 1 bytes/record (0.001 Kbytes)
dt.exe (j:1 t:9):        Total bytes transferred: 20002 (19.533 Kbytes, 0.019 Mbytes)
dt.exe (j:1 t:9):         Average transfer rates: 62 bytes/sec, 0.061 Kbytes/sec, 0.000
Mbytes/sec
dt.exe (j:1 t:9):           Average read latency: 83us (min 25us, max 11.72ms)
dt.exe (j:1 t:9):          Average write latency: 1.84ms (min 33us, max 173.99ms)
dt.exe (j:1 t:9):          Average total latency: 963us (min 25us, max 173.99ms)
dt.exe (j:1 t:9):        Number I/O's per second: 62.064
dt.exe (j:1 t:9):         Number seconds per I/O: 0.0161 (16.11ms)
dt.exe (j:1 t:9):         Total passes completed: 1/1
dt.exe (j:1 t:9):          Total files processed: 20002/20002
dt.exe (j:1 t:9):          Total errors detected: 0/1
dt.exe (j:1 t:9):             Total elapsed time: 05m22.00s
dt.exe (j:1 t:9):                  Starting time: Sun Nov  2 14:51:07 2025
dt.exe (j:1 t:9):                    Ending time: Sun Nov  2 14:56:29 2025
dt.exe (j:1 t:9):

$
```

Please Note: My system disk is a PCIe NVME SSD, thus it is very fast. But that said, the above workload takes much longer running in my Windows Ubuntu VM.

## 4.1 Min/Max Latency Example

```
$ dt of=dt.data bs=random limit=100m runtime=5 latmin=60 latmax=1000 flags=direct
dt.exe (j:1 t:1): Record #8, Write Latency: 1.30ms (slow)
dt.exe (j:1 t:1): Record #13, Write Latency: 7.90ms (slow)
dt.exe (j:1 t:1): Record #14, Write Latency: 2.71ms (slow)
        …
dt.exe (j:1 t:1): Record #2355, Write Latency: 52us (fast)
dt.exe (j:1 t:1): Record #2376, Write Latency: 54us (fast)
        …
dt.exe (j:1 t:1): Record #2219, Read Latency: 1.01ms (slow)
dt.exe (j:1 t:1): Record #2254, Read Latency: 3.29ms (slow)
        …
```

## 4.2 Latency Frequency Examples

```
$ dt of=dt.data bs=random limit=100m runtime=5 latfreq=1000 flags=direct disable=stats
dt.exe (j:1 t:1): Record #1000, Write Latency: 274us
dt.exe (j:1 t:1): Record #1000, Read Latency: 391us
dt.exe (j:1 t:1): Record #2000, Write Latency: 174us
dt.exe (j:1 t:1): Record #2000, Read Latency: 230us
dt.exe (j:1 t:1): The runtime of 5 seconds has expired, terminating thread...
        …
$ dt of=dt.data bs=random limit=100m records=10 latfreq=1 flags=direct disable=stats
dt.exe (j:1 t:1): Record #1, Write Latency: 1.83ms
dt.exe (j:1 t:1): Record #2, Write Latency: 1.67ms
dt.exe (j:1 t:1): Record #3, Write Latency: 707us
dt.exe (j:1 t:1): Record #4, Write Latency: 294us
dt.exe (j:1 t:1): Record #5, Write Latency: 650us
dt.exe (j:1 t:1): Record #6, Write Latency: 2.85ms
dt.exe (j:1 t:1): Record #7, Write Latency: 737us
dt.exe (j:1 t:1): Record #8, Write Latency: 980us
dt.exe (j:1 t:1): Record #9, Write Latency: 756us
dt.exe (j:1 t:1): Record #10, Write Latency: 757us
dt.exe (j:1 t:1): Warning: The bytes written 5790208, is less than the data limit 104857600 requested!
dt.exe (j:1 t:1): Record #1, Read Latency: 420us
dt.exe (j:1 t:1): Record #2, Read Latency: 638us
dt.exe (j:1 t:1): Record #3, Read Latency: 522us
dt.exe (j:1 t:1): Record #4, Read Latency: 193us
dt.exe (j:1 t:1): Record #5, Read Latency: 345us
dt.exe (j:1 t:1): Record #6, Read Latency: 528us
dt.exe (j:1 t:1): Record #7, Read Latency: 378us
dt.exe (j:1 t:1): Record #8, Read Latency: 537us
dt.exe (j:1 t:1): Record #9, Read Latency: 421us
dt.exe (j:1 t:1): Record #10, Read Latency: 285us
$
```