dt v25.02 Release Notes

1. Overview

This document describes the release notes for changes introduced in dt version 25.02:

This is the released versions of dt:

```
root@rtp-smc-qa18-4 ~# dt version
    --> Date: December 3rd, 2021, Version: 25.02, Author: Robin T. Miller <--
root@rtp-smc-qa18-4 ~#
```

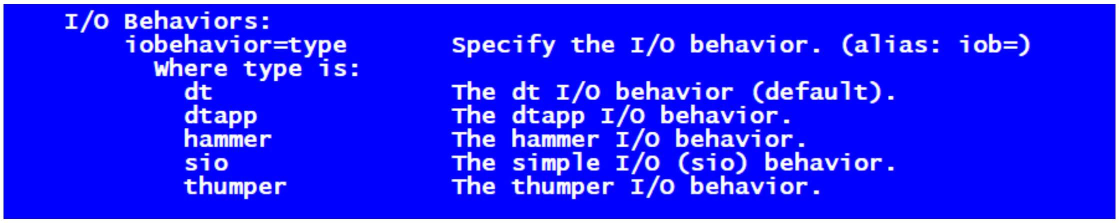
[Note: Sections 2 and 3 removed since they are vendor unique!]

4. New I/O Behaviors

My former colleague at NetApp worked with their legal eagles to allow other dt I/O behaviors to be available as open source!

The additional I/O behaviors are: **dtapp**, **hammer**, and **sio**

This means there are five (5) different tools integrated into dt now:



```
    I/O Behaviors:
        iobehavior=type          Specify the I/O behavior. (alias: iob=)
          Where type is:
            dt                   The dt I/O behavior (default).
            dtapp                The dtapp I/O behavior.
            hammer               The hammer I/O behavior.
            sio                  The simple I/O (sio) behavior.
            thumper              The thumper I/O behavior.
```

4.1. dtapp I/O Behavior

The dtapp I/O behavior was designed to emulate application I/O across a set of disks. Furthermore, those disks can be mirrored (synchronous or asynchronous). In the case of synchronous mirroring, the data is verified immediately, while in the case of asynchronous mirroring, the verification is done later after async data has been written.

Here's a dt regression test script which shows this behavior. I'm using different paths to the same disk for this example. A list of disks are provided, then dtapp will randomly choose a disk to write to, then verify this data against the mirror disk. Note: This dodtapp.dt script is in the dt Scripts/ directory.

```
root@rtp-smc-qa18-4 /usr/local/bin/dt.v25/Scripts# export WRITER="/dev/sdl,/dev/sdm,/dev/sdn"
root@rtp-smc-qa18-4 /usr/local/bin/dt.v25/Scripts# export
MIRROR="/dev/sdam,/dev/sdan,/dev/sdao"
root@rtp-smc-qa18-4 /usr/local/bin/dt.v25/Scripts# dt script=dodtapp
dt> version
--> Date: November 23rd, 2021, Version: 25.01, Author: Robin T. Miller <--
dt>
dt> logprefix="%et %prog %device (j:%job t:%thread): "
dt>
dt> $CAPACITYP=capacityp=10
dt> $LOGDIR=/var/tmp
dt> system rm -f /var/tmp/dtapp_job*.log /var/tmp/dtapp_thread*.log
dt>
dt> $MIRROR_OPTIONS="iobehavior=dtapp bs=random capacityp=10 onerr=abort history=25
keepalivet=30 tag=mirror job_log=/var/tmp/dtapp_job%job.log log=/var/tmp/dtapp_thread-
j%jobt%thread.log"
dt>
dt> $WRITER=/dev/sdl,/dev/sdm,/dev/sdn
dt> $MIRROR=/dev/sdam,/dev/sdan,/dev/sdao
dt>
dt> of=/dev/sdl,/dev/sdm,/dev/sdn mirror=/dev/sdam,/dev/sdan,/dev/sdao iobehavior=dtapp bs=random
capacityp=10 onerr=abort history=25 keepalivet=30 tag=mirror job_log=/var/tmp/dtapp_job%job.log
log=/var/tmp/dtapp_thread-j%jobt%thread.log limit=250m
```

4.2. hammer I/O Behavior

The *hammer* I/O behavior is a file system tool. Hammer performs file system operations, including file locking, delete, rename, and truncate, in addition to writing and reading files. An in-memory list of files is maintained and randomly chosen. By default, hammer runs forever and also supports file system full and file op percentages.

4.2.1. Hammer Help

```
root@rtp-smc-qa18-4 /# dt iobehavior=hammer help

Usage: dt iobehavior=hammer [dir=DirectoryPath] [options...]

    Options:
        -help                      Show this help text, then exit.
        dir=DirectoryPath          Directory path for hammer files.
                                   If omitted, current directory is the
default.
```

```
        -api posix               The only API supported for Unix is POSIX.
        -bg                      Don't use stdin or stdout (output to log
file).
        -checkinodes             File system reusing inodes?
        -filercore=FILERNAME     Coredump the filer upon corruption.
        -nofilercore             Disable coredumping the filer.
        -direct                  Disable filesystem caching.
        -fill                    Fill disk and then keep it full.
        -interactive             Use interactive mode (not supported).
        -iterations NUMBER       The number of iterations to execute.
        -logfile FILE            Use logfile FILE. (Default is none).
        -mode={mixed | creates | overwrites} (Default: mixed)
                                 Use the specified mode.
        -nocleanup               Don't remove files upon completion.
        -noflush                 Don't use flush buffers (write through).
        -onlydelete              When exiting, only delete files.
        -onlyflush               Always use async I/O and flush buffers.
        -runtime=N               Stop hammering after N seconds.
        -threads=value           The number of hammer threads.
        -seed=value              Set the random seed to use.
        -version                 Print the version, then exit.
```

Blocksize Options:

```
        -blocksize=NBYTES        Use blocksize NBYTES.
        -blocksize=random        Use a random blocksize. (Default: True)
        -minbsize=NBYTES         Set minimum block size to NBYTES. (Default:
512)
        -maxbsize=NBYTES         Set maximum block size to NBYTES. (Default:
131072)
        -minfsize=NBYTES         Set minimum file  size to NBYTES. (Default:
1)
        -maxfsize=NBYTES         Set maximum file  size to NBYTES. (Default:
5242880)
```

Error Control Options:

```
        -ignorelockerrors        Don't halt on file locking errors, continue.
        -ignorefileerrors        Don't halt on file operation errors,
continue.
        -ignoredatacorruption    Don't halt on data corruption errors,
continue.
        -ignoreallerrors         Don't halt on any of the above errors,
continue.
                                 [NOTE: hammer will stop on other critical
errors.
                                  that can prevent it from functioning
properly].
```

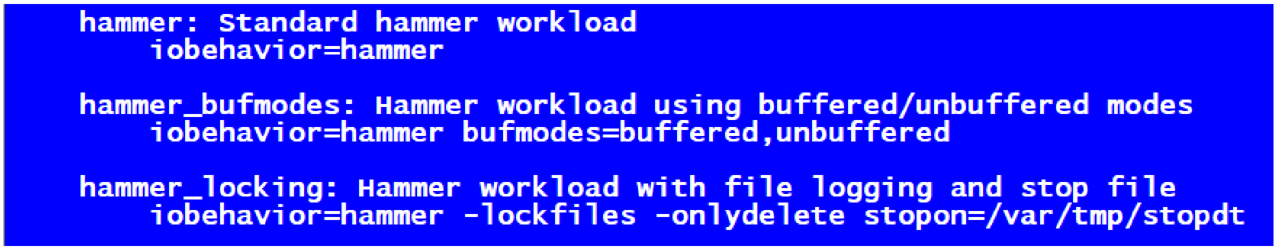
Lock Control Options:

```
        -nolockdebug             Exclude file lock/unlock debug output (it's
chatty).
        -lockfiles               Include file locks (locks & unlocks) using
defaults for the lock options below.
        -lockmode={mixed | full | partial}
                                 More chance of full or partial file locks
(default: mixed).
        -unlockchance=[0-100]    Probability of keeping locks and skipping
unlocking, 0-100 percent.
```

```
        Examples:
           if -unlockchance=100  100% chance of unlocking, ALL files
unlocked. [default]
           if -unlockchance=50    50% chance of unlocking each file.
           if -unlockchance=0      0% chance of unlocking, NO files are
unlocked.

    dt Options Supported:
        bufmodes={buffered,unbuffered,cachereads,cachewrites}
                                 Set one or more buffering modes (Default:
none)
        maxdata=value            The maximum data limit (all files).
        maxdatap=value           The maximum data percentage (range: 0-100).
        maxfiles=value           The maximum files for all directories.
        stopon=filename          Watch for file existance, then stop.
        enable=raw               The read after write flag.

    Also know, I/O monitoring (noprog*= options), keepalive, and trigger=
options
    are also supported with hammer.

Examples:
    % dt iobehavior=hammer dir=/mnt/hammer maxdatap=25 runtime=1h
    % dt iobehavior=hammer dir=/mnt/hammer bufmodes=buffered,unbuffered
stopon=stopfile
    % dt iobehavior=hammer dir=/mnt/hammer -lockfiles -onlydelete -threads=3
log=hammer.log

root@rtp-smc-qa18-4 /#
```

4.2.2. Hammer Workloads

These are the predefined *hammer* workloads:



```
    hammer: Standard hammer workload
        iobehavior=hammer

    hammer_bufmodes: Hammer workload using buffered/unbuffered modes
        iobehavior=hammer bufmodes=buffered,unbuffered

    hammer_locking: Hammer workload with file logging and stop file
        iobehavior=hammer -lockfiles -onlydelete stopon=/var/tmp/stopdt
```

4.2.3. Example Hammer Workload

```
dt (j:1 t:1): Command Line:
dt (j:1 t:1):
dt (j:1 t:1): # ./dt of=/mnt/localhost/iscsi-rtp-smc-qa18-4-v7-4421cc0f0efae5826c9ce900fa42bd6b
workload=hammer sdsf=/dev/mapper/mpathbo logdir=/var/tmp/dtlogs_20211123-142143-300785 job_log=dt-
iscsi-rtp-smc-qa18-4-v7-4421cc0f0efae5826c9ce900fa42bd6b-job%job.log log=dt-iscsi-rtp-smc-qa18-4-v7-
4421cc0f0efae5826c9ce900fa42bd6b-j%jobt%thread.log enable=async onerr=abort stopon=/var/tmp/stopdt
tag=iotag runtime=1m
dt (j:1 t:1):
dt (j:1 t:1): --> Date: November 18th, 2021, Version: 25.01, Author: Robin T. Miller <--
dt (j:1 t:1):
dt (j:1 t:1): Copyright (c) 2012 Network Appliance, Inc. All rights reserved.
dt (j:1 t:1): hammer started at Tue Nov 23 14:21:45 2021
dt (j:1 t:1): version=$Id: hammer.c#10 $
dt (j:1 t:1): path=/mnt/localhost/iscsi-rtp-smc-qa18-4-v7-
4421cc0f0efae5826c9ce900fa42bd6b/52b2c4d5367e759fd296b049c8cc7e9a pid=0x6D49
dt (j:1 t:1): client=rtp-smc-qa18-4 (Linux 3.10.0-1160.2.1.el7.x86_64 #1 SMP Mon Sep 21 21:00:09 EDT 2020
x86_64)
dt (j:1 t:1): minfsize=0x1 maxfsize=0x500000
dt (j:1 t:1): minbsize=0x00000200 maxbsize=0x00020000 blocksize=random
dt (j:1 t:1): api=posix mode=mixed streams=off flush=random nocleanup=false retrydisc=true
seed=7033918714584814846
dt (j:1 t:1): logfile=/var/tmp/dtlogs_20211123-142143-300785/dt-iscsi-rtp-smc-qa18-4-v7-
4421cc0f0efae5826c9ce900fa42bd6b-j1t1.log timezone=EST
dt (j:1 t:1): num_iterations=0 max_iterations=18446744073709551615 cur_runtime=0 max_runtime=60
dt (j:1 t:1): 2021/11/23-14:21:45 CREATE 00000001.ham fileid=0x6B8B4567 blocksize=0x00019E7D
filesize=0x0043503b timestamp=0x619D3F49 (sync) 70992.8K/s
dt (j:1 t:1): 2021/11/23-14:21:45 TRUNC 00000001.ham fileid=0x6B8B4567 oldsize=0x0043503b
newsize=0x000f0f42 0.000987sec
dt (j:1 t:1): 2021/11/23-14:21:45 CREATE 00000002.ham fileid=0x327B23C6 blocksize=0x000061CD
filesize=0x00244c28 timestamp=0x619D3F49 (sync) 28500.7K/s
dt (j:1 t:1): 2021/11/23-14:21:46 CREATE 00000003.ham fileid=0x643C9869 blocksize=0x00002B87
filesize=0x00062eca timestamp=0x619D3F4A (sync) 13948.2K/s
dt (j:1 t:1): 2021/11/23-14:21:46 TRUNC 00000002.ham fileid=0x327B23C6 oldsize=0x00244c28
newsize=0x00101ca7 0.000356sec
dt (j:1 t:1): 2021/11/23-14:21:46 CREATE 00000004.ham fileid=0x66334873 blocksize=0x00010927
filesize=0x0015d12a timestamp=0x619D3F4A (async then flush) 262411K/s
dt (j:1 t:1): 2021/11/23-14:21:46 TRUNC 00000001.ham fileid=0x6B8B4567 oldsize=0x000f0f42
newsize=0x00083dda 0.000167sec
dt (j:1 t:1): 2021/11/23-14:21:46 TRUNC 00000001.ham fileid=0x6B8B4567 oldsize=0x00083dda
newsize=0x00078f0e 4e-05sec
dt (j:1 t:1): 2021/11/23-14:21:46 READ 00000001.ham fileid=0x6B8B4567 blocksize=0x00014314
filesize=0x00078f0e 842794K/s
dt (j:1 t:1): 2021/11/23-14:21:46 DELETE 00000001.ham fileid=0x6B8B4567 0.000178sec
dt (j:1 t:1): 2021/11/23-14:21:46 TRUNC 00000003.ham fileid=0x643C9869 oldsize=0x00062eca
newsize=0x00040804 6.3e-05sec
dt (j:1 t:1): 2021/11/23-14:21:46 RENAME 00000003.ham fileid=0x643C9869 newpath=00000005.ham 3.3e-05sec
```

4.3. sio I/O Behavior

The *sio* I/O behavior is a performance tool. As a performance tool, no data verification is performed by default. Performance statistics are reported upon completion.

4.3.1. sio Help

```
root@rtp-smc-qa18-4 /# dt iobehavior=sio help
```

```
Usage: dt iobehavior=sio [options...]

sio (Simple I/O Load Generator) - NetApp
A tool to generate artificial I/O workloads against any device
Supports numerous configuration variables (reads vs writes, etc)
Supports multiple devices and multiple threads. Collects a wide
variety of statistics on I/O client machines and/or I/O servers.
```

Basic Usage:

```
dt iobehavior=sio readp=<read%> randp=<rand%> bs=<blksz> starting=<start> \
       end=<end> runtime=<secs> threads=<threads> devs=<dev>,[devs,...]
readp=<read>          Percentage of accesses that are reads. Range [0,100].
                      'random' keyword makes the read/write percentage
random.
                      BEWARE, writing to a file is unchecked and will trash
files.
randp=<rand>          Percentage of acceses that are random. Range [0,100].
                      Sequential accesses = 0%, else random percentage
bs=<blksz>            Size of I/O's. Example: 2k, 4k, 1m
                      'random' keyword makes the I/O size random 512 bytes to
262144 bytes.
ibs=<blksz>           Size of read requests. (overrides bs= option)
obs=<blksz>           Size of write requests.
start=<strt_byte>     Lower bound for access location in each file.
end=<file_size>       Total bytes accessed in each file (e.g. 100m, 2g,
1000k).
runtime=<seconds>     Runtime for test. Counting starts AFTER all threads
have started.
threads=<numthreads>  Concurrent I/O generators. Uses real individual
threads.
dev=<dev>             Device to access. May be file (foo.out) or device
(/dev/dsk/etc).
 or devs=<dev>[,...]  Multiple devices and/or files can be specified, comma
separated.
 or file=<paths>      One or more paths to files to access (synonym for
'devs' option).
```

Examples:

```
 1) Random 4k I/O with 25% reads/75% writes, 75% random/sequential for 10
minutes.
    Accessing a total of 250 megabytes in each file, after prefilling the
file.
    % dt iobehavior=sio file=a.file,b.file bs=4k readp=25 randp=75 end=250m -
prefill runtime=10m
 2) Random reads and writes with random block sizes via 10 threads to the
same file.
    This test will run infinitely without -numops or runtime options.
    % dt iobehavior=sio file=a.file bs=random readp=random end=1g -direct -
verify threads=100
```

Options:

```
    Options are divided into four categories:
        Basic Features, Advanced Features,
        Q/A Features, and Esoteric Stuff.
```

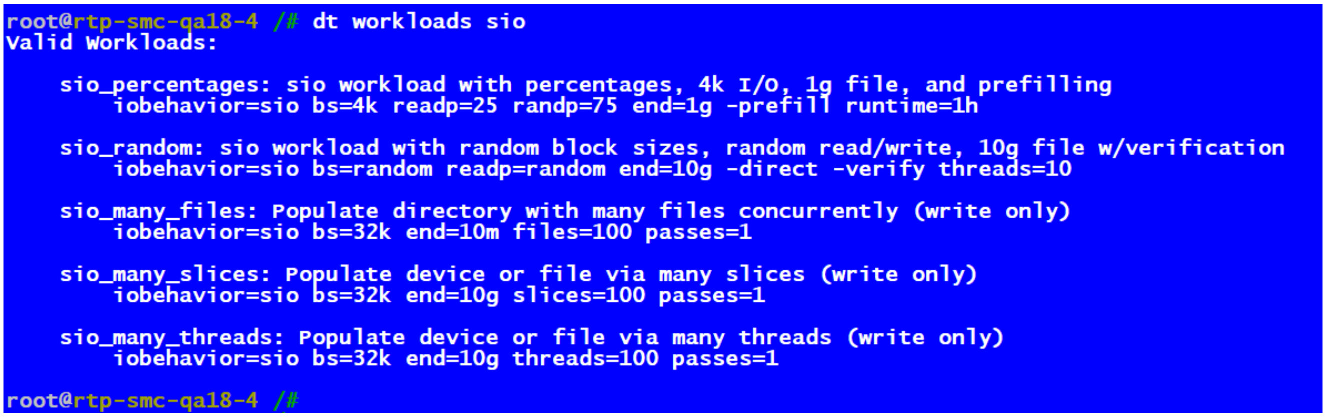
Basic Features:

```
    -help                 List this sio help, then exit.
    -version              Display this detailed version log.
    -noflock              Do NOT lock files.  Locking affects caching on some
OS's.
    -noheader             Suppress single line header output. (Good for
multiple runs).
    -debug                Output detailed debug info. Be prepared for a lot
of info.
```

```
    -Debug               Very verbose debug information. Be prepared for a
lot of info!
    -niceoutput          Print output in single column, human-readable
format.
    -prettyprint         Pretty print the output (this is dt's format).
    -no_dsync            Do NOT open files with O_DSYNC. Allows async
writes.
    -noperf              Do NOT display performance statistics.

Advanced Features:
    -stop=<fname>        Watch for existance of file 'fname' and terminate.
    -think=<msec>        Each thread waits 'ms' MS before issuing each I/O.
    -iops=<rate>         Target IOPS for each thread.
    -lockall             Lock the complete file as opposed to a single byte.
    -truncate:           IFF pure sequential writes, then when I/O wraps to
                         beginning of file, the file is truncated.
    -max_blksize=<bytes> Set maximum block size to 'bytes'.
    -max_latency=(ms)    Maximum allowed latency (in milliseconds) of an IO.
    -direct              Disable filesystem caching.
    -align=(size)        Alignment to be used with random block size.
    -break_on_dc         Exit upon detecting data corruption ASAP.

Q/A Features:
    -verify              Read back written data and verify content.
    -verify_retry=<n>    Retry failed verifies 'n' times.
    -instrumentation     Special pattern insertion technique.
    -fixedfill=<value>   Fill the file with 8 bit value.

Esoteric Stuff:
    -numops=<num_ops>    Run for 'num_ops' I/O's and stop. Beware stats.
    -fileperthread       Open one file per thread. Special names.
    -blockno             Prints out the I/O block numbers.
    -iofailok            Allow I/O failures (do not access file again).
    -iomutex             Use mutex to synchronize multiple threads.
    -fillonce            Write all files once, then stop.
    -prefill             Write all files prior to test I/O.
    -partition_among_threads Partition the file among threads.

root@rtp-smc-qa18-4 /#
```

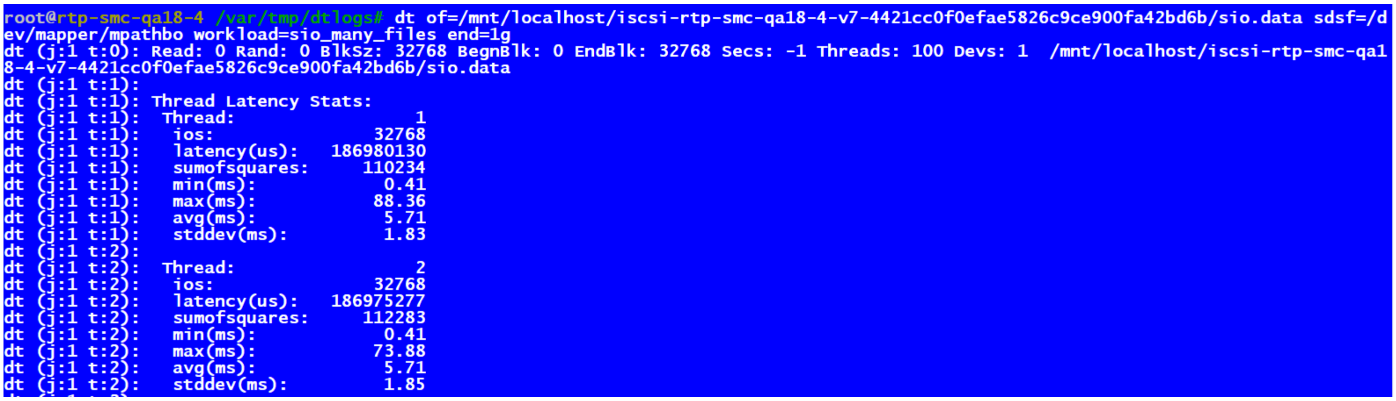
sio Workloads

These are the predefined *sio* workloads:



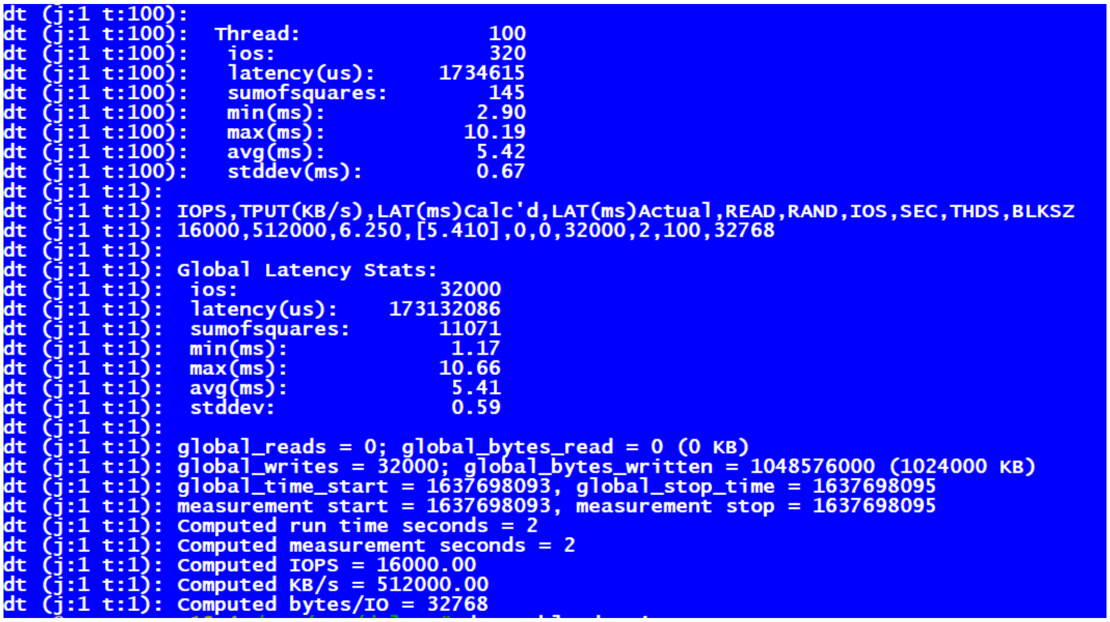
```
root@rtp-smc-qa18-4 /# dt workloads sio
Valid Workloads:

    sio_percentages: sio workload with percentages, 4k I/O, 1g file, and prefilling
        iobehavior=sio bs=4k readp=25 randp=75 end=1g -prefill runtime=1h

    sio_random: sio workload with random block sizes, random read/write, 10g file w/verification
        iobehavior=sio bs=random readp=random end=10g -direct -verify threads=10

    sio_many_files: Populate directory with many files concurrently (write only)
        iobehavior=sio bs=32k end=10m files=100 passes=1

    sio_many_slices: Populate device or file via many slices (write only)
        iobehavior=sio bs=32k end=10g slices=100 passes=1

    sio_many_threads: Populate device or file via many threads (write only)
        iobehavior=sio bs=32k end=10g threads=100 passes=1

root@rtp-smc-qa18-4 /#
```

4.3.2. Example sio Workload



```
root@rtp-smc-qa18-4 /var/tmp/dtlogs# dt of=/mnt/localhost/iscsi-rtp-smc-qa18-4-v7-4421cc0f0efae5826c9ce900fa42bd6b/sio.data sdsf=/d
ev/mapper/mpathbo workload=sio_many_files end=1g
dt (j:1 t:0): Read: 0 Rand: 0 BlkSz: 32768 BegnBlk: 0 EndBlk: 32768 Secs: -1 Threads: 100 Devs: 1  /mnt/localhost/iscsi-rtp-smc-qa1
8-4-v7-4421cc0f0efae5826c9ce900fa42bd6b/sio.data
dt (j:1 t:1):
dt (j:1 t:1): Thread Latency Stats:
dt (j:1 t:1):  Thread:                1
dt (j:1 t:1):   ios:              32768
dt (j:1 t:1):   latency(us):  186980130
dt (j:1 t:1):   sumofsquares:    110234
dt (j:1 t:1):   min(ms):           0.41
dt (j:1 t:1):   max(ms):          88.36
dt (j:1 t:1):   avg(ms):           5.71
dt (j:1 t:1):   stddev(ms):        1.83
dt (j:1 t:2):
dt (j:1 t:2):  Thread:                2
dt (j:1 t:2):   ios:              32768
dt (j:1 t:2):   latency(us):  186975277
dt (j:1 t:2):   sumofsquares:    112283
dt (j:1 t:2):   min(ms):           0.41
dt (j:1 t:2):   max(ms):          73.88
dt (j:1 t:2):   avg(ms):           5.71
dt (j:1 t:2):   stddev(ms):        1.85
```

Full output emitted.



```
dt (j:1 t:100):
dt (j:1 t:100):  Thread:                100
dt (j:1 t:100):   ios:                  320
dt (j:1 t:100):   latency(us):      1734615
dt (j:1 t:100):   sumofsquares:         145
dt (j:1 t:100):   min(ms):             2.90
dt (j:1 t:100):   max(ms):            10.19
dt (j:1 t:100):   avg(ms):             5.42
dt (j:1 t:100):   stddev(ms):          0.67
dt (j:1 t:1):
dt (j:1 t:1): IOPS,TPUT(KB/s),LAT(ms)Calc'd,LAT(ms)Actual,READ,RAND,IOS,SEC,THDS,BLKSZ
dt (j:1 t:1): 16000,512000,6.250,[5.410],0,0,32000,2,100,32768
dt (j:1 t:1):
dt (j:1 t:1): Global Latency Stats:
dt (j:1 t:1):  ios:               32000
dt (j:1 t:1):  latency(us):    173132086
dt (j:1 t:1):  sumofsquares:       11071
dt (j:1 t:1):  min(ms):             1.17
dt (j:1 t:1):  max(ms):            10.66
dt (j:1 t:1):  avg(ms):             5.41
dt (j:1 t:1):  stddev:              0.59
dt (j:1 t:1):
dt (j:1 t:1): global_reads = 0; global_bytes_read = 0 (0 KB)
dt (j:1 t:1): global_writes = 32000; global_bytes_written = 1048576000 (1024000 KB)
dt (j:1 t:1): global_time_start = 1637698093, global_stop_time = 1637698095
dt (j:1 t:1): measurement start = 1637698093, measurement stop = 1637698095
dt (j:1 t:1): Computed run time seconds = 2
dt (j:1 t:1): Computed measurement seconds = 2
dt (j:1 t:1): Computed IOPS = 16000.00
dt (j:1 t:1): Computed KB/s = 512000.00
dt (j:1 t:1): Computed bytes/IO = 32768
```