MDTest Benchmark Rules for SBCC 2026

Abir Haque

Rules and Constraints

The MDTest rules for SBCC are a modified version of the rules for the IO500: `https://io500.org/rules/submission`.

1. Submissions are made using the latest version of MDTest application within the IOR GitHub: `https://github.com/hpc/ior`.
2. Read-after-write semantics: The system must be able to correctly read freshly written data from a different client node after the close operation on the writer has been completed.
3. The stonewall flag must be set to 300 to ensure all create/write run for at least 300 seconds.
4. There can be no edits made to the source code.
5. The file names for the mdtest output files for the benchmarking phase may not be pre-created, i.e., the benchmark directory must be completely empty prior to execution.
6. You must run all phases of the benchmark on a single storage system without interruption.
7. There is no limitation on the number of storage nodes, the storage servers may optionally be co-located on the client nodes.
8. All data must be written to persistent storage within the measured time for the individual benchmark, e.g. if a file system caches data, it must ensure that data is persistently stored before acknowledging the close.
9. Data and metadata must be written in its entirety and not reduced based on its contents.
10. Only submissions using at least 4 physical client nodes are eligible and at least one benchmark process must run on each client.
11. Teams must submit proof that they are not using non-persistent storage for their submission, such as the output of `df -T <target>`, where `<target>` is the directory where the benchmark is executed.

Tasks

The MDTest tasks for SBCC are a modified version of the tasks for the IO500: `https://io500.org/about` (description), `https://github.com/IO500/io500/tree/main/src` (specific commands).

1. Easy – 4 Nodes

A. **Write:** Each process creates a unique directory and creates zero-byte files within that directory at the maximum rate possible. Each process creates the same number of files as the rank that created the most files when the 300s stonewall timer is hit.

B. **Stat:** Each process fetches the file attributes of the files created by a process on a different node during the write phase.

C. **Delete:** Each process deletes all of the files and directory created by a process on a different node during the write phase.

Command: To execute the previously stated tasks in the proper order, you must have the following flags enabled in the correct order at the very least for an acceptable submission. Additional commands/flags that don't affect the completion of the required phases are acceptable (e.g. using MPI for distributed execution). You must provide your benchmark results output (so the graders can calculate the score), evidence of `dir` being persistent storage (`df -T <dir>` at the very least), and stonewall file output.

```
mdtest -n <N> -u -L -F -P -N 1 -d <dir> -x <stonewall_file> -Y -W 300 -w 0
 -C -T -r
```

2. Easy – Hero Run, $\geq$ 4 Nodes

Execute the same commands from Task 1 with the exception that you are allowed to use more than 4 nodes. **You are allowed to reuse your output from Section 3 if you are unable to perform larger run for whatever reason.**

A. **Write:** See 1.A.

B. **Stat:** See 1.B.

C. **Delete:** See 1.C.

Command: To execute the previously stated tasks in the proper order, you must have the following flags enabled in the correct order at the very least for an acceptable submission. Additional commands/flags that don't affect the completion of the required phases are acceptable (e.g. using MPI for distributed execution). You must provide your benchmark results output (so the graders can calculate the score), evidence of `dir` being persistent storage (`df -T <dir>` at the very least), and stonewall file output.

```
mdtest -n <N> -u -L -F -P -N 1 -d <dir> -x <stonewall_file> -Y -W 300 -w 0
 -C -T -r
```

3. Hard – 4 Nodes

A. **Write:** A single directory is created and each process creates 3901-byte files with a unique name within that directory at the maximum rate possible. Each process creates the same number of files as of the rank which was able to write the most data within the 300s stonewall time period.

B. **Stat:** Each process fetches the file attributes of the files created by a process on a different node during the write phase.

C. **Read:** Each process reads the 3901-byte data of the files created by a process on a different node during the previous write/stat phases.

D. **Delete:** Each process deletes all of the files created by a process on a different node during the previous write/stat/read phases.

Command: To execute the previously stated tasks in the proper order, you must have the following flags enabled in the correct order at the very least for an acceptable submission. Additional commands/flags that don't affect the completion of the required phases are acceptable (e.g. using MPI for distributed execution). You must provide your benchmark results output (so the graders can calculate the score), evidence of `dir` being persistent storage (`df -T <dir>` at the very least), and stonewall file output.

```
mdtest -n <N> -t -w 3901 -e 3901 -P -N 1 -F -d <dir> -x <stonewall_file> -
C -T -E -r -W 300 -Y
```

4. Hard – Hero Run, $\geq$ 4 Nodes

Execute the same commands from Task 3 with the exception that you are allowed to use more than 4 nodes. **You are allowed to reuse your output from Section 3 if you are unable to perform larger run for whatever reason.**

A. **Write:** See 3.A.

B. **Stat:** See 3.B.

C. **Read:** See 3.C.

D. **Delete:** See 3.D.

Command: To execute the previously stated tasks in the proper order, you must have the following flags enabled in the correct order at the very least for an acceptable submission. Additional commands/flags that don't affect the completion of the required phases are acceptable (e.g. using MPI for distributed execution). You must provide your benchmark results output (so the graders can calculate the score), evidence of `dir` being persistent storage (`df -T <dir>` at the very least), and stonewall file output.

```
mdtest -n <N> -t -w 3901 -e 3901 -P -N 1 -F -d <dir> -x <stonewall_file> -
C -T -E -r -W 300 -Y
```

Grading

The score for each sub-task is the maximum operations/second for each phase (14 total sub-tasks/phases). The final score will be calculated via a geometric mean of the scores obtained from each sub-task (tree creation/removal will be ignored):

$$\text{Score} = \sqrt[14]{1.\text{A} * 1.\text{B} * 1.\text{C} * 2.\text{A} * 2.\text{B} * 2.\text{C} * 3.\text{A} * 3.\text{B} * 3.\text{C} * 3.\text{D} * 4.\text{A} * 4.\text{B} * 4.\text{C} * 4.\text{D}}$$

For any term not submitted, the term score will automatically be 1.

Deliverables

For each task, provide execution/setup scripts, benchmark results output (so the graders can calculate the score), evidence of `dir` being persistent storage (`df -T <dir>` at the very least), and the stonewall file. All information must be provided to ensure reproducibility.

Questions

Contact Abir Haque (`abirhaque@gatech.edu`) for any questions/clarifications regarding the benchmark rules.