# Introduction

With every new file system we want to assess the performance in terms of maximum and scaling. This document describes the tests to run to determine these answers. It will cover both data write and read as well as metadata. These tests can be run at any point, so they are not specific to a certain percentage full or fragmentation level.

## Data Write and Read

These tests measure the scaled performance of writing and reading data for the file system. We’re interested in weak scaling, strong scaling, timed I/O, and timed maximum I/O. In all cases we are interested in MPI/IO and POSIX modes for N-1 and N-N workloads.

### Weak Scaling

These tests are reflective of how an application might use the file system because the total amount of data written or read is generally dependent on the number of processes the application uses. They are not completely indicative of how an application might fare with the file system because we use a few large I/Os to transfer more data per I/O than an application would probably use.

Each pe does a total I/O of 1 GiB in 50 MiB chunks.

#### IOR Command for MPI/IO and N-1

ior -a MPIIO -b 50m -C -g -o <filename> -k -r -s 20 -t 50m -w

#### IOR Command for POSIX and N-1

ior -a POSIX -b 50m -C -g -o <filename> -k -r -s 20 -t 50m -w

#### IOR Command for MPI/IO and N-N

ior -a MPIIO -b 1g -C -F -g -o <filename> -k -r -t 50m -w

#### IOR Command for POSIX and N-N

ior -a POSIX -b 1g -C -F -g -o <filename> -k -r -t 50m -w

### Strong Scaling

These tests show how many processes it takes to maximize performance for a given data volume. We'll use a data volume equal to 1 GiB \* max-pes-in-experiment, TD (total data). These tests will show us the num-pes that gets the best performance for a given data volume that would be generated by an experiment that would typically be spread out over more processes.

#### IOR Command for MPI/IO and N-1

ior -a MPIIO -b 50m -C -g -o <filename> -k -r -s TD/num-pes -t 50m -w

#### IOR Command for POSIX and N-1

ior -a POSIX -b 50m -C -g -o <filename> -k -r -s TD/num-pes -t 50m -w

#### IOR Command for MPI/IO and N-N

ior -a MPIIO -b TD -C -F -g -o <filename> -k -r -t 50m -w

#### IOR Command for POSIX and N-N

ior -a POSIX -b TD -C -F -g -o <filename> -k -r -t 50m -w

### Timed I/O

This is done to put enough data through the system at each stage of scaling that the file open and close overheads do not dominate. But, the I/O is done where every process writes or reads, which is the default configuration for most applications. We need to tell IOR to write SMD (so much data) that it can't finish in 3 minutes (180 seconds, see -D parameter).

#### IOR Command for MPI/IO and N-1

ior -a MPIIO -b 50m -C -D 180 -g -o <filename> -k -r -s SMD/50-MiB -t 50m -w

#### IOR Command for POSIX and N-1

ior -a POSIX -b 50m -C -D 180 -g -o <filename> -k -r -s SMD/50-MiB -t 50m -w

#### IOR Command for MPI/IO and N-N

ior -a MPIIO -b SMD -C -D 180 -F -g -o <filename> -k -r -t 50m -w

#### IOR Command for POSIX and N-N

ior -a POSIX -b SMD -C -D 180 -F -g -o <filename> -k -r -t 50m -w

### Timed Maximum I/O

This is done, again, to put enough data through the system at each stage of scaling that the file open and close overheads do not dominate. But, the I/O is done where the processes that write or read are constrained to be optimal number for the given file system and we distribute them over different node counts. We have to know how many writers and readers are optimal for the given file system. Other than the distribution of a fixed number of processes over different node counts with the "mpirun" command for the system, the IOR commands themselves are the same as for Timed I/O, but we have to define SMD (so much data) to be enough data that the jobs can't finish in 3 minutes (180 seconds, see -D parameter).

#### IOR Command for MPI/IO and N-1

ior -a MPIIO -b 50m -C -D 180 -g -o <filename> -k -r -s SMD/50-MiB -t 50m -w

#### IOR Command for POSIX and N-1

ior -a POSIX -b 50m -C -D 180 -g -o <filename> -k -r -s SMD/50-MiB -t 50m -w

#### IOR Command for MPI/IO and N-N

ior -a MPIIO -b SMD -C -D 180 -F -g -o <filename> -k -r -t 50m -w

#### IOR Command for POSIX and N-N

ior -a POSIX -b SMD -C -D 180 -F -g -o <filename> -k -r -t 50m -w

## Metadata

Metadata tests are done in order to see how the file system performs in relation to file creation, file stat, and file deletion. The process count is scaled to determine where max performance occurs. Four classes of metadata tests are defined below:

#### mdtest command for create, stat, and delete one million files in a single directory

aprun –n <#pes> -N <#pes-per-node> ./mdtest -n <1000000/#pes> -d <path-to-pfs>/<nn\_shared-dir> -F -C -T -r -N<#pes-per-node>

#### mdtest command for create, stat, and delete one million files in X directories

where X is the number of metadata targets (MDTs) in a DNE-enabled Lustre file system. mdtest uses the –M option to create on each of the metadata targets.

aprun –n <#pes> -N <#pes-per-node> ./mdtest -n <1000000/#pes> -d <path-to-pfs>/<nn\_dne-dir> -F -C -T -r -N -M<#pes-per-node>

#### mdtest command for create, stat, and delete one million files in a directory per process configuration.

aprun –n <#pes> -N <#pes-per-node> ./mdtest -n <1000000/#pes> -d <path-to-pfs>/<nn\_unique-dir> -F -C -T -r -N <#pes-per-node> -u

#### mdtest command for each process creates, stats, and deletes the same file.

aprun –n <#pes> -N <#pes-per-node> ./mdtest -S -C -T -r -n 1 -d <path-to-pfs>/<n1\_shared-dir> -F