MPI-IO Test User's Guide
Version 1.000.21
October 4, 2008

1. **Quick Start**

```
tar xvfz mpi_io_test_21.tar.gz
cd mpi_io_test/test_fs/src
make clean all
./mpirun -np 1 ./fs_test.x -type 1 -size 65536 -nobj 1 \
-target /path/to/file
```

2. **Introduction**

Although there are a host of existing file system and I/O test programs available, most were not designed with parallel I/O in mind and are not useful at the scale of our clusters. Los Alamos National Lab's MPI-IO Test was written with parallel I/O and scale in mind. The MPI-IO test is built on top of MPI's I/O calls and is used to gather timing information for reading from and writing to file(s) using a variety of I/O profiles; N processes writing to N files, N processes writing to one file, N processes sending data to M processes writing to M files, or N processes sending data to M processes to one file. A data aggregation capability is available and the user can pass down MPI-IO, ROMIO and some file system specific MPI hints.

By default, the MPI-IO Test will write a specific pattern to a file, close the file, open the file for read, read the data, check for data integrity and close the file. Timing information is reported for file open, close, read and write data rates, and effective bandwidths.

The MPI-IO Test can be used for performance benchmarking and, in some cases, to diagnose problems with file systems or I/O networks.

3. **Running The Benchmark**

The user is allowed to input program variables on the command line or as an input file (-setup). If a set up file is provided, it is read first and then all command line variables are read in. This way, the user can have a master file with all standard inputs, but vary one or a small number of variables by specifying them on the command line and, thus, overwriting the variables in the setup file.

The user can provide MPI info hints that are loaded into the MPI_info stucture by using the "-hints" flag. Any hint can be input to the routine, such as standard MPI -IO hints; striping_factor, striping_width, cb_buffer_size, and cb_nodes; ROMIO specific hints; start_iodevice, ind_rd_buffer_size, ind_wr_buffer_size, direct_read, and direct_write; and file system specific hints are recognized. According to the MPI-2 standard, an implementation of MPI-IO may choose to silently ignore any hint. Thus, any hint that is entered and not recognized by MPI-IO will be silently discarded. Although there is overlap between some of the program inputs and MPI-IO hints, such as the input routine parameter "-num_io" and the MPI-IO hint "-cb_nodes", a hint will be loaded into the MPI info structure only if the hint flag is used, i.e. the -num_io parameter will

not trigger the hint cb_nodes to be set. Thus, it is up to the user to specify the correct hints and routine parameters for the desired test.

For data aggregation, processes are chosen to perform I/O based on if the user supplies a list of hosts to do I/O (-host). If a host list exists, and a number of I/O processors (-num_io) is specified for that host, say P, then the first P processes on that host will do I/O. Any processes on that host that does not do I/O, i.e. any processes beyond the first P, will send data to an I/O process on its own host. If there are processes on hosts that do not perform I/O, they will send data to an I/O process on an I/O host. This situation can lead to an imbalance of I/O processes per host. If the user does not specify what hosts will do I/O and P processes are specified to do I/O, then every process with rank a multiple of (P - 1) will do I/O. Thus, non-I/O processes with rank K will send data to the I/O processor with the greatest rank not exceeding K.

When sending more than one object (-nobj), memory is allocated for just one object and that same object is sent -nobj times. In order to get around objects being sent from cache, the user is allowed to specify if the object is touched (-touch) before being sent. The user can choose to:

(1) not touch the object at all, meaning the same object is sent -nobj times with no modification,

(2) touch every object once, meaning the byte offset of the object in the file is written to the first element of each object or

(3) touch every object once per page, meaning the byte offset of the object in the file is written to the first element of each page in the object.

In order to run the N to M test, "-type" 3, 4, 5, …

4. Input Parameters

The following flags are command-line input parameters for the MPI-IO Test routine with brief descriptions of each of the flags. The "#" represents a number and "%" represents a string input.

-type # Type of data movement:
(1) N processors to N files
(2) N processors to 1 file

-collective Use collective read and write calls (MPI_File_read/write_at_all) instead of independent calls (MPI_File_iread/iwrite). (Default: Use independent read/write calls)

-nobj # Number of objects to write and read to/from file.

-strided # All processors write in a strided, all processors write to one region of the file and then all seek to the next portion of the file, or non-strided, all processors have a region of the file that they exclusively write into, pattern.
0. Non-strided (Default)
1. Strided - not implemented for use with aggregation (-type 3 and 4)

-size # Number of bytes that each object will send to the I/O processors or write to file.

-*target %* Full path and file name to write and read test data. The following in the file name or path will resolve to:

%r processor rank
%h host name
%p processor ID
%s Unix epoch

-*deletefile* Causes the target file to be deleted at the end of the program. (Default: FALSE)

-*touch #* Specifies to write a particular pattern into the data file. The pattern is written for only some of the bytes in the file depending on the argument:

(0) Never (default)
(1) Once per object
(2) Once per page per object
(3) Once per byte per page per object (i.e. every byte)

-*check #* When reading the data file, check that every element equals the expected value, which depends on the "touch" option. The values are the same as for touch above.

-*sync* Sync the data before file close on writes (Default: FALSE - do not sync data)

-*sleep #* Each processor will "sleep" for this number of seconds between when the file is closed from writing and opened to read. (Default: 0 seconds)

-*barriers %* When specified, processors will do a barrier as specified. The argument is a comma separated list of [b|a][open|close|read|write]. Example –barriers aopen,bwrite means to do a barrier after every open and before every write.

-*hints %s=%s[,%s=%s]* Multiple MPI-IO key value hint pairs. Any MPI, ROMIO or file system specific hint can be specified here.

-*op %* (Optional) Only read or write the target file.

1. "read" - only read the target files
2. "write" - only write out the target files

-*shift* If specified, this makes each process read different data than what it wrote in an attempt to avoid caching affects. Each process will shift its “rank” halfway around the rank space.

-*time #* If specified, this limits how many seconds the program will run. If you specify a time for an N-1 nonstrided IO pattern, you must also specify a –supersize (below).

-*supersize #* If run with a time limit and an N-1 nonstrided IO pattern, the user must specify a supersize value. Nonstrided means that each process has its own non-shared region of the file. But this only makes sense if the amount of data written by each process is predetermined. As its not in a time limit case, use supersize to try to guess at what it should be. If the region is filled,

then the process will jump to the next region (i.e. it will create a hybridization of N-1 strided and nonstrided IO patterns).

-*help* Display the input options

5. Return Values

The MPI-IO test will return 0 on success and -1 otherwise.

6. Hints

Since the user is allowed to input any key, value pair using the "-hints" command line parameter, any MPI-IO, ROMIO, or file system specific hints can be used. Since the MPI-2 standard allows for hints to be silently rejected if they are not recognized, input hints may not be recognized by MPI and no warning will be issues. For this reason, if a hint is specified by the user, all hints will be printed out after the write phase of the test is run.

For a full list of hints specified by the MPI-2 standard, please consult a reference on MPI-IO or the MPI-2 standard. The following is a list of hints that may be useful for I/O:

striping_factor - Hint specifying the number of I/O devices across which the file should be striped
striping_unit - Hint specifying the number of consecutive bytes of a file that are stored on a particular I/O device
cb_buffer_size - Hint specifying the size of the temporary buffer that can be used on each processor for collective I/O
cb_nodes - Hint specifying the number of processor that should actually perform collective I/O

The following are ROMIO specific hints.
cb_config_list - Hint specify what hosts and how many processors on those hosts perform I/O

start_iodevice = Hint specifying the I/O device from which file striping should begin.

ind_rd_buffer_size - Hint specifying the temporary buffer ROMIO uses for data sieving.

ind_wr_buffer_size - Hint specifying the temporary buffer ROMIO uses for data sieving.

Some hints specific to the MPI implementation of Quadrics MPI:

serialize_open – Serialize the call to POSIX open in MPI_File_open().

Some hints specific to the Panasas file system; PanFS:

panfs_concurrent_write - When set to 1, PanFS concurrent write mode is enabled.

panfs_raid_group_enable - When set to 1, files are created using PanFS raid groups.

panfs_raid_group_stripe_unit - The size of the stripe unit in bytes for the PanFS raid group.

panfs_raid_group_stripe_width - The number of Storage Blades in a PanFS raid group.

panfs_raid_group_stripe_depth - The number of stripes stored contiguously in a PanFS raid group.

panfs_raid_group_total_blades - The total number of Storage Blades in which a file is striped. The # of raid groups will be (panfs_raid_group_total_blades /panfs_raid_group_stripe_width)

7. Database integration via mysql

The test supports dumping results into a mysql database for later querying. The results contain the timing values (the same that are printed to the stdout), the input parameters, and various such environmental information as can be obtained such as version information for the file system (if it is Panasas), hostnames of the processes, operating system information, etc.

Various environment variables control whether the test is built with the mysql integration:

a. MYSQL_HOST: If set, this is the hostname where the mysql server is running. This assumes that the name of the database is mpi_io_test and the table is named experiment and that authentication is completely open.
b. MYSQL_FILE: If set instead of MYSQL_HOST, this is the path to a file where the test will write out queries. This file can be later uploaded to a mysql server.
c. MPI_INC: If the MYSQL_HOST variable is set, then the Makefile must be instructed about how to compile with the mysql libraries. In addition to using the standard Makefile variables, MPI_INC can also be used to pass information about how to find the mysql include files.
d. MPI_LD: Similarly, if the MYSQL_HOST variable is set, the MPI_LD variable can be used to provide link information about the mysql libraries to the Makefile.
e. MY_MPI_HOST: This value can be set to the fullpath to the MPI implementation being used. Doing so allows comparisons between different MPI versions.

8. Examples

a. SAMPLE COMMAND LINE

==========

Example 1:

```
mpirun -np 50 fs_test.x -type 2 –strided 1 \
-nobj 64 -size 131072  \
-target /scratch/jnunez/test_file_t2 \
-touch 2 –check 2 \
-hints panfs_concurrent_write=1
```

9. Timing Results

All times reported have a minimum, average and maximum time that any process took to complete the operation. The following are the times that are reported to the output file:

Effective Bandwidth: The aggregate bandwidth in Megabytes per second for reading or for writing data including time to open and close the output data file. Computed as: (number processors * object size * number of objects * sizeof(double))/ (total elapsed time) / (1024.0 * 1024.0)

Total Time: Elapsed time, including time to open and close the output data file, to complete the read or write phase.

Write (or Read) Bandwidth: The aggregate bandwidth in Megabytes per second for reading or for writing data excluding time to open and close the output data file. Computed as: (number processors * object size * number of objects * sizeof(double))/ (total elapsed time - time to open file – time to close file) / (1024.0 * 1024.0)

Write (or Read) Time: Elapsed time, excluding time to open and close the output data file, to complete the read or write phase.

MPI File Open Time: Time to open the output data file(s).

MPI File write (or read) wait time: Amount of time any I/O processor took waiting to write to the output data file(s). Since the independent read and write calls are non-blocking, this is time a process waited for previous write to complete. Since the collective calls are blocking, this value is just the time it takes to complete the collective read or write call.

MPI File Preallocate Time: Amount of time it took to preallocate the file.

MPI File Close Wait Time: Amount of time it took to close the output data file(s).

MPI File Sync Wait Time: Amount of time it took to sync the output data file(s).

MPI Processor send wait time = The maximum time any one processor took to send all data objects. Only valid for tests using aggregation.

MPI Processor receive wait time = The maximum time any one processor waited to receive all data objects. Only valid for tests using aggregation.

a. SAMPLE OUTPUT

==========

Sample Output:

```
Input Parameters:
I/O Testing type N -> N: 1
Total number of processors (N): 96
Total number of I/O processors (M): 96
Host Name List: None
Number of circular buffers (-ahead): 0
```

```
Number of objects (-nobj): 106
Number of bytes in each object (-size): 4194304
Circular buffer size (-csize): 4194304
Preallocate file size (-preallocate): -1
Check data read (-chkdata): 3
Use collective read/write calls (-collective): 0
Flush data before file close (-sync): 0
Strided data layout (-strided): 0
Delete data file when done reading (-deletefile): 1
Write output data to file (-nofile): 1
Aggregators send data to themselves (-norsend): 1
Test data written to (-target): /panfs/REALM1/scratch/jnunez/test_4194304_106.0
Touch object option (-touch): 3
Timing results written to (-output):
/home2/jnunez/Panasas/results/client2.3.0/tests_20050822/nn_p96_s4194304_o106_r1
Errors and warnings written to (-errout): stderr

Environment Variables (MPI):
MPIRUN_DEVICE=ch_p4

=== MPI write [96->96->96 /panfs/REALM1/scratch/jnunez/test_4194304_106.0]
Starting ...
=== Effective Bandwidth (min [rank] avg max [rank]): 1.734188e+03 [80]
2.026611e+03 2.817814e+03 [0] Mbytes/s.
=== Total Time (min [rank] avg max [rank]): 1.444524e+01 [0] 2.008476e+01
2.347150e+01 [80] sec.
=== Write Bandwidth (min [rank] avg max [rank]): 2.007613e+03 [80] 2.386687e+03
3.121080e+03 [0] Mbytes/s.
=== Write Time (min [rank] avg max [rank]): 1.304164e+01 [0] 1.705460e+01
2.027482e+01 [80] sec.
=== MPI File Open Time (min [rank] avg max [rank]): 3.811100e-02 [0]
2.135186e+00 2.518727e+00 [37] sec.
=== MPI File Write Wait Time (min [rank] avg max [rank]): 4.060000e-04 [95]
1.628927e-03 2.344900e-02 [27]  sec.
=== MPI File Preallocate Time (min [rank] avg max [rank]): 2.000000e-06 [15]
2.437500e-06 3.000000e-06 [1] sec.
=== MPI File Close Wait Time (min [rank]  avg max [rank]): 6.816700e-02 [57]
8.949620e-01 2.347709e+00 [73]  sec.
=== Completed MPI-IO Write.
=== MPI read [96->96->96 /panfs/REALM1/scratch/jnunez/test_4194304_106.0]
Starting ...
=== Effective Bandwidth (min [rank] avg max [rank]): 1.955103e+03 [28]
2.800417e+03 6.659021e+04 [95] Mbytes/s.
=== Total Time (min [rank] avg max [rank]): 6.112610e-01 [95] 1.453498e+01
2.081937e+01 [28] sec.
=== Read Bandwidth (min [rank] avg max [rank]): 1.955134e+03 [28] 2.800652e+03
6.756456e+04 [0] Mbytes/s.
=== Read Time (min [rank] avg max [rank]): 6.024460e-01 [0] 1.453376e+01
2.081903e+01 [28] sec.
=== MPI File Open Time (min [rank] avg max [rank]): 2.760000e-04 [28] 1.134844e-
03 4.264400e-02 [0] sec.
=== MPI File Read Wait Time (min [rank] avg max [rank]): 3.690000e-04 [95]
2.950417e-03 2.887000e-02 [43]  sec.
=== MPI File Preallocate Time (min [rank] avg max [rank]): 0.000000e+00 [0]
0.000000e+00 0.000000e+00 [0] sec.
=== MPI File Close Wait Time (min [rank]  avg max [rank]): 4.900000e-05 [95]
7.812500e-05 7.680000e-04 [6]  sec.
```

```
=== Completed MPI-IO Read.
```

MAKEFILES
=========
Makefiles have been provided for Linux, Tru64, and IRIX64 operating systems.

ALSO INCLUDED
=============

10. CONTACT

=======
Please report any bugs to
James Nunez (jnunez@lanl.gov) at Los Alamos National Labs
Please address any questions about the mysql integration to
John Bent (johnbent@lanl.gov) at Los Alamos National Labs