Los Alamos National Lab MPI-IO Test

User's Guide

Version 1.0

Open Source Software

Released Under LA-CC-05-013

Last Updated: March 2, 2006

Copyright

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.

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, to be added.

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
(3) N processors to M processors to M files
(4) N processors to M processors to 1 file
(5) N processors to M files

-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)

-num_io "# # ... #" Total number of I/O processors. If using the -host flag, this is the number of I/O processors on each host. For type 5, this is the number of files to write to. (Default: all N processors will do I/O)

-nobj # Number of objects to write to file or to send to each of the M I/O processors (Default: 1)

-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. Must be a multiple of sizeof(double). Allows for (Linux) dd-like expressions; w 2, b 512, KB 1000, K 1024, MB 1,000,000, M 1,048,576, GB 1,000,000,000, and G 1,073,741,824. A positive number followed by one of the above letter will result in the two multiplied, i.e. 2b => 1024.

-*csize #* Size of each I/O processors circular buffers

-*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
(Default: ./test_file.out)

-*targets %* File name of file containing one file name for each processor. File should have one file name on each line and an IO processor with IO rank N will read the file name from line
N+1. The special characters for -target flag are valid. Test type 5 must use this flag and the
first M (-num_io) file names will be read from the targets file.

-*deletefile* Delete the target file(s) at the end of the program. (Default: FALSE)

-*barrier* Call MPI_Barrier before every write call. (Default: FALSE)

-*touch #* All data objects sent/written by a processor will contain the sequence of (double) values from processor rank to (size input vairable + processor rank). The first value can be overwritten with the following flags:

(1) Never (default)
(2) Once per object
(3) Once per page per object

-*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)

-*chkdata #* When reading the data file, check that every element equals the expected value, which depends on the "touch" option. Check the following data elements:

(0) None
(1) First data element of each object (Default)
(2) First two data elements of each object
(3) All data elements

-preallocate # Preallocate this number of bytes before write.

-*ahead #* Number of circular buffers for asynchronous I/O (Default: 0)

-*host "% % ... %"* List of host names to do I/O (Default: all hosts)

-*lhosts* Print out all hosts running this program.

-*dio* Use direct read and direct write for all read and write operations. This flag will evoke the ROMIO specific hints "direct_read" and "direct_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 %* Read or write the target file. If this flag is not used, both write then read will be performed.
"read" - only read the target files [Not implemented yet]
"write" - only write out the target files

-*nofile* Only allocate memory and send message, do not write the messages to file.

-*norsend* In normal aggregation operation, all processors send data to an IO processor. Even IO processors send data to themselves. This flag indicates that IO processors do not send data to themselves.

-*output %* File to write timing results, input parameters, and MPI environment variables. (Default: stdout)

-*errout %* File to write all errors encountered while running the program. (Default: stderr)

-*verbose* Print out all times for each process. (Default: Off)

-*help* Display the input options

Return Values

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

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

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

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

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

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 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.

SAMPLE COMMAND LINES

Example 1: 50 processes write to a single file with 20480 64 Kbyte messages. A unique value is written to each page of the message and the data is check during the read process.

mpirun -np 50 mpi_io_test.x -type 2 -nobj 20480 -size 65536 -target /scratch/jnunez/test_file -touch 3 –chkdata 3 -hints striping_factor 4 cb_nodes 6 -output ./output_n1.txt -deletefile

Example 2: 144 processes each writing to their own file with 8192 64 Kbyte messages. A unique value is written to each page of the message and the data is check during the read process.

mpirun –np 144 mpi_io_test.x -type 1 -nobj 8192 -size 65536 -target /scratch/jnunez/test_file.%r -touch 3 –chkdata 3 -output ./output_nn.txt -deletefile

SAMPLE OUTPUT

The following output is from the example 2 above.

```
Input Parameters:
I/O Testing type N -> N: 1
Total number of processors (N): 144
Total number of I/O processors (M): 144
Host Name List: None
Number of circular buffers (-ahead): 0
Number of objects (-nobj): 8192
Number of bytes in each object (-size): 65536
Circular buffer size (-csize): 65536
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): /scratch/jnunez/test_file.0
Touch object option (-touch): 3
Timing results written to (-output): /home2/jnunez/output_nn.txt
Errors and warnings written to (-errout): stderr

Environment Variables (MPI):
LAMPI_ADMIN_AUTH0=27066
LAMPI_ADMIN_AUTH1=16837
LAMPI_ADMIN_AUTH2=1140249739
LAMPI_ADMIN_PORT=59925

=== MPI write [144->144->144 /panfs/REALM1/raid0/jnunez/raid5/test_65536_8192.0]
 Starting ...
=== Effective Bandwidth (min [rank] avg max [rank]): 1.780664e+03 [69]
2.070630e+03 2.659093e+03 [0] Mbytes/s.
=== Total Time (min [rank] avg max [rank]): 2.772675e+01 [0] 3.560656e+01
4.140479e+01 [69] sec.
=== Write Bandwidth (min [rank] avg max [rank]): 1.988787e+03 [140] 2.353562e+03
3.151956e+03 [90] Mbytes/s.
=== Write Time (min [rank] avg max [rank]): 2.339119e+01 [90] 3.132614e+01
3.707185e+01 [140] sec.
=== MPI File Open Time (min [rank] avg max [rank]): 9.945100e-02 [0]
2.371871e+00 6.845830e+00 [112] sec.
=== MPI File Write Wait Time (min [rank] avg max [rank]): 2.077227e+01 [90]
2.880994e+01 3.437072e+01 [140]  sec.
```

```
=== MPI File Preallocate Time (min [rank] avg max [rank]): 1.000000e-06 [124]
1.125000e-06 2.000000e-06 [119] sec.
=== MPI File Close Wait Time (min [rank]  avg max [rank]): 3.256570e-01 [37]
1.908540e+00 3.916407e+00 [39]  sec.
=== Completed MPI-IO Write.
=== MPI read [144->144->144 /panfs/REALM1/raid0/jnunez/raid5/test_65536_8192.0]
Starting ...
=== Effective Bandwidth (min [rank] avg max [rank]): 1.699342e+03 [26]
3.206577e+03 5.739535e+03 [0] Mbytes/s.
=== Total Time (min [rank] avg max [rank]): 1.284564e+01 [0] 2.299274e+01
4.338620e+01 [26] sec.
=== Read Bandwidth (min [rank] avg max [rank]): 1.719073e+03 [26] 3.267071e+03
5.969634e+03 [0] Mbytes/s.
=== Read Time (min [rank] avg max [rank]): 1.235051e+01 [0] 2.256700e+01
4.288824e+01 [26] sec.
=== MPI File Open Time (min [rank] avg max [rank]): 3.275100e-02 [128]
2.286173e-01 3.446270e-01 [115] sec.
=== MPI File Read Wait Time (min [rank] avg max [rank]): 1.014155e+01 [19]
2.074131e+01 4.040744e+01 [26]  sec.
=== MPI File Preallocate Time (min [rank] avg max [rank]): 0.000000e+00 [143]
0.000000e+00 0.000000e+00 [0] sec.
=== MPI File Close Wait Time (min [rank]  avg max [rank]): 1.220660e-01 [116]
1.971161e-01 3.624860e-01 [32]  sec.
=== Completed MPI-IO Read.
```

CONTACT

Please report any bugs to
James Nunez (jnunez@lanl.gov) at Los Alamos National Labs