IOR: I/O Performance Benchmark

Summary Version
1.0

Purpose of Benchmark
IOR is used for testing performance of parallel filesystems using various interfaces and access patterns.

Characteristics of Benchmark
IOR uses MPI for process synchronization.

Location of Benchmark

Mechanics of Building Benchmark

- The officially maintained version of IOR is available on sourceforge: http://sourceforge.net/projects/ior-sio
  We are using the 2.10.1 version of the benchmark.
- Type 'gmake [posix|mpiio|hdf5|ncmpi|all]' from the IOR/ directory.
- In IOR/src/C, the file Makefile.config currently has settings for AIX, Linux, OSF1 (TRU64), and IRIX64 to model on.
- Note that MPI must be present for building/running IOR, and that MPI I/O must be available for MPI I/O, HDF5, and Parallel netCDF builds
- HDF5 and Parallel netCDF libraries are necessary for those builds.
- All IOR builds include the POSIX interface.

Mechanics of Running Benchmark

Instructions for running IOR are given in the USER_GUIDE.
More specific examples of how one might run the specified tests follow:

1) **Small problem: IOR command line example:**

```
srun –N1 –n1 -ppdebug ./IOR.exe \
-Cge -vv -wWr -F –i5 -t 256k -b1m  -o /p/lscratcha/rhedges/testFile
```

This example runs one process on 1 nodes.
The transfer size, block size, and segment count can be adjusted and the segment count times the block size is the amount of data written by each process.

2) **Medium problem**

```
srun –N512 –n2048 -ppdebug ./IOR.exe \
-Cge -vv -wWr -F –i5 -t1m –b16m –s4 –o /p/lscratcha/rhedges/testFile
```

This example runs 4 processes per node on 512 nodes in a file per process mode. Each process writes 64Mbytes, or 256 Mbytes per node.

3) Large Sequoia problem

a) Scalable Science

```
i) srun –N96k –n96k -ppdebug ./IOR.exe \
--Cge vv -wWr -F –i5 –t4m –b1g -o /p/lscratcha/rhedges/testFile
```

one task per compute node writes a 1Gbyte file.

```
ii)   srun –cpus-per-task=16 –N96k -ppdebug ./IOR.exe \
-Cge -vv -wWr -F –i5 –t4m –b128m -o /p/lscratcha/rhedges/testFile
```

all 16 tasks on a compute node write their own 64 Mbyte files for a total of 1Gbyte per compute node.

b) Throughput

```
i) srun –N4k –n4k -ppdebug ./IOR.exe \
-Cge -vv -wWr -F –i5 –t4m –b1g -o /p/lscratcha/rhedges/testFile
```

one task per compute node writes a 1Gbyte file.

```
ii)   srun –cpus-per-task=16 –N4k -ppdebug ./IOR.exe \
-Cge -vv -wWr -F –i5 –t4m –b128m -o /p/lscratcha/rhedges/testFile
```

all 16 tasks on a compute node write their own 64 Mbyte files for a total of 1Gbyte per compute node.

4) CORAL class problems:

- *All nodes involved in tests*
- *All processors involved in the file or writer per process tests.*
- *The transfer size can be adjusted by the vendor to yield optimal throughput*
- *The data per node should equal the amount of memory on a node*

a) File per node:

```
srun –Nnumber_compute_nodes  -ppdebug ./IOR.exe \
--Cge vv -wWr -F –i5 –txfer_size –bdata_per_node  \
-o  /p/lscratcha/rhedges/testFile
```

b) File per process:

```
srun –Nnumber_compute_nodes  \
–nnumber_compute_nodes*processes_per_node   \
-ppdebug ./IOR.exe --Cge vv -wWr -F –i5 \
–txfer_size –bdata_per_node/processes_per_node  \
-o  /p/lscratcha/rhedges/testFile
```

c) Shared File, one writer per node, rank data contiguous:

```
srun –Nnumber_compute_nodes  -ppdebug ./IOR.exe \
--Cge vv -wWr –i5 –txfer_size –bdata_per_node  \
-o  /p/lscratcha/rhedges/testFile
```

d) Shared File, one writer per process, rank data interleaved, with 16 segments from each process:

```
srun –Nnumber_compute_nodes  \
–nnumber_compute_nodes*processes_per_node   \
-ppdebug ./IOR.exe \
--Cge vv -wWr –i5 –txfer_size  -s16 \
–bdata_per_node/(processes_per_node*16) \
-o  /p/lscratcha/rhedges/testFile
```

5) IOR input script example (optional)

IOR can also take input parameter from a file/script. The following is such an example

launch as : `srun -N2 -n16 -ppdebug ./IOR -f  IOR.input`

IOR.input for this example is the following:

```
===============> start script <===============
IOR START
  api=POSIX
  testFile=/p/lscratcha/rhedges/testFile
  repetitions=5
  readFile=1
  writeFile=1
  filePerProc=1
  checkWrite=0
  checkRead=0
  keepFile=0
  segmentCount=1
  blockSize=1g
  transferSize=1m
  reorderTasks=0
  deadlineForStonewalling=100
  useExistingTestFile=0
RUN
```

The following IOR.input lists all parameters that can be set in a script

```
==============> start script <==============
IOR START
  api=POSIX
  testFile=/p/crater/rhedges/testFile
  repetitions=5
  multiFile=0
  interTestDelay=5
  readFile=1
  writeFile=1
  filePerProc=1
  checkWrite=1
  checkRead=1
  keepFile=1
  quitOnError=0
  segmentCount=1
  blockSize=32k
  outlierThreshold=0
  transferSize=32k
  singleXferAttempt=0
  individualDataSets=0
  verbose=0
  numTasks=8
  collective=1
  preallocate=0
  useFileView=0
  keepFileWithError=0
  setTimeStampSignature=0
  useSharedFilePointer=0
  useStridedDatatype=0
  uniqueDir=0
  fsync=0
  storeFileOffset=0
  maxTimeDuration=60
  deadlineForStonewalling=0
  useExistingTestFile=0
  useO_DIRECT=0
  showHints=0
  showHelp=0
RUN
```

Example IOR.input files for file per process, and for shared files with data interleaved or not are provided on the CORAL web site.

Verification of Results

1) The FOM reported is the throughput where timing is from the first file open to last file closed.

2) Correctness of the data is verified in test 1). In particular the flag “W” causes read of the written data, and a comparison to the data that was written.