IO-500

John Bent, Julian Kunkel, Jay Lofstead

Seagate, DKRZ, SNL

Why do we need an IO 500?

Storage researchers need an IO 500

Users need an IO 500

"I run my job on a new machine and I get X GB/s. Is that expected? I google the machine and I see that the storage is rated at 1 TB/s. Do I need to spend time tuning my IO?"

Admins need an IO 500

"Users always ask me what IO patterns they should use for good performance. I have no idea."

Procurers need an IO 500

"I wrote a bad benchmark specification in my RFP and ended up with a bad storage system."

What will the IO 500 accomplish?

Bound expectations

A mix of the best and the worst IO patterns

Force honesty

Best and worst numbers are published for all to see

Create balanced focus on metadata and data

HPC has been over-emphasizing bandwidth for too long

Discover best practices and share with entire community

Submitters configure their own parameters for best

These parameters and other tuning are published

Today's Two Topics

1. Benchmark Rules and Process
   a. Specific tests and parameters
   b. Submission rules
   c. Scoring
2. Community Rules and Process
   a. Committee membership
   b. Decision making process

Top 500 Lists key requirements

- Easy to run
  - not a lot of different tests
  - can be run in under an hour (ideally)
  - easy to configure tests

- Single number for initial comparison
  - but keep detailed information for more nuanced comparison

IO-500 Proposal

- Use IOR and mdtest in an “easy” and “hard” setup
  - Easy is site chosen to best reflect potential optimal cases—but parameters chosen must be shared so others can examine what is done
  - Hard is pre-defined to really tax the system (unoptimized applications)
- Then do a ‘find’ operation
  - Custom tools are fine so long as functionality is the same
- At least 5 minute run time for for the write/create phase
  - Then how ever long read/stat takes
- Must use the **same number** of ranks for all tests
- Prefer users to submit results for each storage subsystem
  - e.g., burst buffers and PFS are two separate runs

Single consolidated LANL github

Nathan Hjelm has merged mdtest and IOR into LANL github

No more #ifdef's in mdtest!

They share the very nice IOR function pointer table

Add a new backend just once and it works for both

https://github.com/IOR-LANL/ior

```
Signed-off-by: Nathan Hjelm <hjelmn@lanl.gov>
```

hjelmn committed 13 hours ago 1 parent 4a27a2a commit 4240bc37725728f6ed73e0845d5dafdb4923d740

Showing **3 changed files** with **11 additions** and **2 deletions**. Unified | Split

2 META View

```
@@ -1,3 +1,3 @@
 Package: ior
-Version: 3.0.1
+Version: 3.0.2
 Release: 0
```

9 NEWS View

```
@@ -0,0 +1,9 @@
+IOR NEWS
+========
+
+Last updated 2017-06
+
+3.0.2
+
+- IOR and mdtest now share a common codebase. This will make it easier
+run performance benchmarks on new hardware.
```

Processing Steps

- Do all 4 write tests and then all 4 read tests to ensure flushed caches
  - IOR easy
  - mdtest easy
  - IOR hard
  - mdtest hard
  - ‘find’ operation

- Consumers consume data that was produced on *different* nodes

```
# produce phase
bndw1 = ior    easy write
iops1 = mdtest easy create
timer = get time
bndw2 = ior    hard write
iops2 = mdtest hard create

# consume phase
bndw3 = ior    easy read
iops3 = mdtest easy stat
bndw4 = ior    hard write
iops4 = mdtest hard stat

# find phase
iops5 = find . -name \*00\* -newer timer

# calculate score
bndw  = geo_mean( bndw1 bndw2 bndw3 bndw4 )
iops  = geo_mean( iops1 iops2 iops3 iops4 iops5 )
score = bndw * iops
```

Open Questions / TODO

1. IOR hard is N-1 strided or N-1 random offsets
   a. If random, validate it works. Might need to add a -random_seed parameter.
2. mdtest needs to shift reader ranks
   a. IOR has this built-in, mdtest does not
3. Run mixed workloads concurrently
   a. Need this for both IOR and mdtest?
   b. Do this within IOR/mdtest or simply by running two concurrent MPI jobs?

Proposed webpage at http://io500.org (hosted by Virtual Institute for IO)

Site	System	Score	Bandwidth (TB/s)	IOPs (Millions)	Capacity (PB)	Submission Date
LANL	Trinity - BB	6.82	3.1	2.2	4	08/08/2017
LANL	Trinity - PFS	.99	0.9	1.1	80	08/08/2017

Show More Entries Advanced Search Show More Columns

Advanced search allows filtering and sorting on arbitrary operations on arbitrary columns (e.g. bandwidth1/capacity)
Other metrics that we hope to include in full view are media type (e.g. SSD), RAID level (e.g. erasure), etc.

Community Rules and Process

Initial Steering Committee

- John Bent, Julian Kunkel, Jay Lofstead
- Decisions based on consensus after open discussion on mailing list
- After IO500 is established and running smoothly, committee membership can be revisited by full community

Benchmark Submission and Rules

- Use the community github repo's for benchmark codes
- For 'easy' tests, can modify benchmark to add custom backend but must submit all modifications with results
- Submit results along with all necessary reproducability information

Might Change Metrics in Future

- Initial proposal for md-real-io instead of mdtest
  - Offers more of a workflow-centric model
  - Need to generate peer-reviewed publications to establish this as a new, standard benchmark (targeting PDSW-DISCS@SC17 and IPDPS 2018)
  - Need to see this requested in more RFP's
- Some ask for application IO kernels representative of different applications
  - Since such variety, IOR is the general representative
  - For sites with predominantly specific workloads, adding IO kernel results can be used to advertise (or re-sort list) for users to determine which platform is the best.
- Need to incorporate these into the metrics so that users can sort.
  - Initial list will be IOR and mdtest since they are widely trusted.

Resulting number

- Geometric mean of the 4 bandwidths
  - Total data size is determined during write
- Geometric mean of the 5 IOPs
  - Total metadata entries determined during create phase
- Product of these two numbers
  - total size is determined during write
  - total metadata entries determined through create phase
- Example for DKRZ:
    - [Need the numbers here to show]

BACKUP SLIDES

mdtest parameters

- easy is a 0 byte file
  mdtest -n *my_test* -u -L -N 1 -b 1 -e 0 -i 1 -w 0
- hard is 3900 byte file (avoid data to be stored on metadata server)
  mdtest -n *my_test* -u -L -N 1 -b 1 -e 3900 -i 1 -w 3900
  -n *my_test* is how much to create and unlink in 10 minutes total.
  -N 1 adds the stride of 1 process between tests
  e.g., assumes cyclic placement of processes.
  -L files only at leaf mode
  -u one unique directory per process (-b 1)
  -e -w controls bytes

IOR parameters for HARD

setup
mpirun -np *my_test* -ppn 1 ./IOR -A 1 -a MPIIO -vv -w -F -u -o data e -k -b 8192M -t 4K -i 60 -m

duplex read test:

- mpirun –np *my_test* –ppn 1 ./IOR -A 1 -a MPIIO -vv -w -F -u -o data_bg -z -k -b 65536M -t 4K -i 30
- mpirun –np *my_test* –ppn 1 ./IOR -A 1 -a MPIIO -vv -r -F -u -o data -z -k -b 8192M –t 4K -i 60 –m

duplex write test:

- mpirun –np *my_test* –ppn 1 ./IOR -A 1 -a MPIIO -vv -r -F -u -o data_bg -z -k -b 65536M -t 4K -i 30 –m
- mpirun –np *my_test* –ppn 1 ./IOR -A 1 -a MPIIO -vv -w -F -u -o data -z -k -b 8192M -t 4K -i 60 –m

Detailed Run Instructions

- N-1 strided 47KiB
  - ior -write {HARD} -fileset f1
- 3900 byte files, single directory
  - md-test -write {HARD} -fileset f2
- User's choice (but must specify what is done)
  - ior -write {EASY} -fileset f3
- 0 byte files, 1 per directory
  - md-test -write {EASY} -fileset f4

Detailed Run Instructions

- N-1 strided 47KiB
  - ior -read {HARD} -fileset f1
- 3900 byte files, single directory
  - md-test -read {HARD} -fileset f2
- User's choice (but must specify what is done)
  - ior -read {EASY} -fileset f3
- 0 byte files, 1 per directory
  - md-test -read {EASY} -fileset f4

md-real-io Parameters

md-real-io -O=1 -m=1000 -D=1 -P=10000 -- -D=/tmp/test

-O=1 shift rank by one

-m limit free memory per node to 1 GByte

-P Number of files to precreate

-D=1 use one directory per rank

if to use the phases, use -1 (for create) -2 (for test) or -3 (for delete)

Some Early Results on KAUST

md-real-io -O=1 -D=1 -P=1000 -- -D=<Burst Buffer!>

1024 nodes: 19,000 obj/s 141.6 Mib/s

- Creates: 50,000/s, 185 MiB/s
- Deletes: 60,000/s

md-real-io -S=$((100*1024*1024)) -O=1 -D=1 -I=10 -P=10

- 3000 obj/s 600 GByte/s !

mdtest

- numbers on 1024 nodes: (Georgios explains): 17689, 69758, 52093, 70653