

Configure, Tune, and Benchmark a Lustre Filesystem

John Fragalla

Principal Architect

High Performance Computing

Agenda

- Server Side Benchmarking
- Lustre Tuning Parameters and Striping
- Client Side Benchmarking using IOR and MDTEST

Lustre Server Side Tuning and Benchmarking

Server Side Benchmark

- Using obdfilter-survey is a Lustre benchmark tool that measures OSS and backend OST performance and does not measure LNET or Client performance
- This is a good benchmark to isolate network and client from the server
- Must run as root to execute obdfilter-survey on the OSS nodes.

obdfilter-survey Parameters

- Example of obdfilter-survey parameters

```
[root@oss1 ~]# nobjlo=1 nobjhi=1 thrlo=256 thrhi=256 size=65536 obdfilter-survey
```

- Parameters Defined

- size=65536 // file size per OST (2x Controller Memory is good practice)
- nobjhi=1 nobjlo=1 // number of files per OST
- thrhi=256 thrlo=256 // number of worker threads per OST

- If you see results significantly lower than what is expected, rerun the test multiple times to ensure those results are not consistent.
- This benchmark can also target individual OSTs if we see an OSS node performing lower than expected, it can be because of a single OST performing lower due to drive issue, RAID array rebuilds, etc.

```
[root@oss1 ~]# targets="fsname-OST0000 fsname-OST0002" nobjlo=1 nobjhi=1 thrlo=256 thrhi=256 size=65536 obdfilter-survey
```

Lustre Tuning Guidelines

xyratex.

Client Lustre Parameters

- **Disable Network Checksums (15% Performance Improvement)**
 - Default is turned on and impacts performance. Disabling this improves performance
- **Max RPCs in Flight (15% Performance Improvement)**
 - RPC is remote procedure call and indicates how much traffic is introduced on LNET per OST from the client
 - Default is 8, Increase to 32 for IB, and up to 256 for Ethernet, in some cases
 - Depends on number of clients
 - This tunable is the maximum number of concurrent RPCs in flight from clients.
- **Max Dirty MB (Can Improve Read Performance)**
 - Default is 32, good rule of thumb is 4x the value of `max_rpcs_in_flight`.
 - Defines the amount of MBs of dirty data can be written and queued up on the client

Ethernet Tuning

- Jumbo Frames has a $\geq 30\%$ improvement on Lustre Performance compared to standard MTU of 1500
- Change MTU on Client and Servers to 9000
- Change MTU on the Switches to 9214 (or max MTU size) to accommodate for payload overhead

Lustre Striping

- Default Lustre Stripe size is 1M and stripe count is 1
 - Each file is written to 1 OST with a stripe size of 1M
 - When multiple files are created and written, MDS will do best effort to distribute the load across all available OSTs
- The default stripe size and count can be changed. Smallest Stripe size is 64K and can be increased by 64K and stripe count can be increased to include all OSTs
 - Changing stripe count to all OSTs indicates each file will be created using all OSTs. This is best when creating a single shared file from multiple Lustre Clients
- One can create multiple directories with various stripe sizes and counts to optimize for performance

Client Side Benchmark

Using IOR and MDTEST

IOR and MDTEST

- IOR and MDTEST use MPI to execute the tests across multiple Lustre/Compute Clients
- SSH keys are required to setup per client to allow remote execution
- IOZONE Benchmark not covered but can discuss later over Coffe

Measuring Performance using IOR

- Within IOR, one can configure the benchmark for File-Per-Process, and Single-Shared-File
 - File-Per-Process: Creates a unique file per task and most common way to measure peak throughput of a Lustre Parallel Filesystem
 - Single-Shared-File: Creates a Single File across all tasks running on all clients
- Two primary modes for IOR
 - Buffered: This is default and takes advantage of Linux page caches on the Client
 - DirectIO: Bypasses Linux page caching and writes directly to the filesystem

IOR Rule of Thumb

- Always want to transfer 2x the memory size of the total number of clients used to avoid any client side caching effect
- In our example:
 - $(200_Clients * 32GB) * 2 = 12800GB$
 - Total file size for the IOR benchmark will be 12800GB
 - NOTE: Typically all nodes are uniform.

Defining IOR Parameters

- Typical IOR Parameter for 200 nodes with 32GB of memory is

```
/usr/lib64/openmpi/bin/mpirun -machinefile machinefile.txt -np 800 --byslot./IOR -v -F -t 1m  
-b 8g -o /mnt/lustre/test.`date+"%Y%m%d.%H%M%S"``
```

- -np 800 = all 200 nodes used with 4 slots (tasks) per node
- -b 16g = $(2 \times 32\text{GB} \times 200 \text{ Clients}) / 800 \text{ tasks}$
- -o /mnt/lustre/test.`date +"%Y%m%d.%H%M%S"``
 - Found using different output test files provides better performance than reusing the same filename for each run
- -F : File per Process (removing this flag will result in single shared file)
- -t 1m: File transfer size of 1m
- -v : verbose output
- -D <time_seconds> - Deadline for Stonewalling
- -B : Direct IO option, bypass any client page caching

Segments in IOR

- Using both segment count and block size options to define total transfer size per task, In some cases this method using a block size of 4mb or 8mb, and a larger segmentation count, will reduce the lock contention that is present when just using a large block size in the range of “gb”. Using the smaller block size, matching the directory stripe size to equal that of the block size within the IOR command; demonstrated a matched performance for all tasks running across all 32 physical clients.
 - 32 Clients against 128 OSTs: configure a stripe count of 128 and only use 32 client in the IOR Run, but use the segment -s flag
 - `lfs setstripe -c 128 -s 4M /mnt/lustre/share`
 - `/usr/lib64/openmpi/bin/mpirun -machinefile machinesfile.txt -np 384 --bynode ./IOR -v -b 4m -t 4m -s 8192 -C -o /mnt/lustre/share/test.out`

MDTEST

- Good benchmark to measure Metadata Operations per Second
- Recommend using a Lustre output directory if stripe count of 1 and stripe size of 1m
- Typical MDTEST Options
 - -v : Verbose output
 - -F : Perform tests on files only and used to test File Operations Per Second
 - -u Unique working directory for each task (Typically Used when only wanting to measure File Operations per Second)
 - -z, -b, -D : Specify depth of the Directory and number of branches, and test only Directories and used to measure Directory Operations per Second
- Recommend to create at least 1 Million Files to remove client caching effect

Conclusion

xyratex.

Conclusion

- There are many configuration parameters and options to change or tune to optimize performance for Lustre
- Stripe size and stripe count play an important role in benchmarking Lustre
- In addition to Lustre options, IOR and MDTEST also have many options and flags that can be used to optimize the benchmark to measure throughput or operations
- Benchmarking requires patience, creativity, and time

References

xyratex.

References

- IEEE “Optimizing Performance of HPC Storage Systems”
 - Torben Kling-Petersen and John Fragalla, Xyratex
 - <http://goo.gl/0M0HCd>
- Lustre Manuals
 - <http://goo.gl/EeNzW>
 - <http://goo.gl/Y2bjx>
 - <http://goo.gl/pO6r0>
- IOR
 - <http://goo.gl/ctfmD>
 - <http://goo.gl/aD6fA>
- obdfilter-survey
 - <http://goo.gl/2TfTO>

Configure, Tune, and Benchmark a Lustre Filesystem

John_Fragalla@xyratex.com

Thank You