DataDirect™ NETWORKS

Lustre File System Acceleration Using Server or Storage-Side Caching: Basic Approaches and Application Use Cases

Presented by **James Coomer**

Senior Technical Advisor

April, 2014

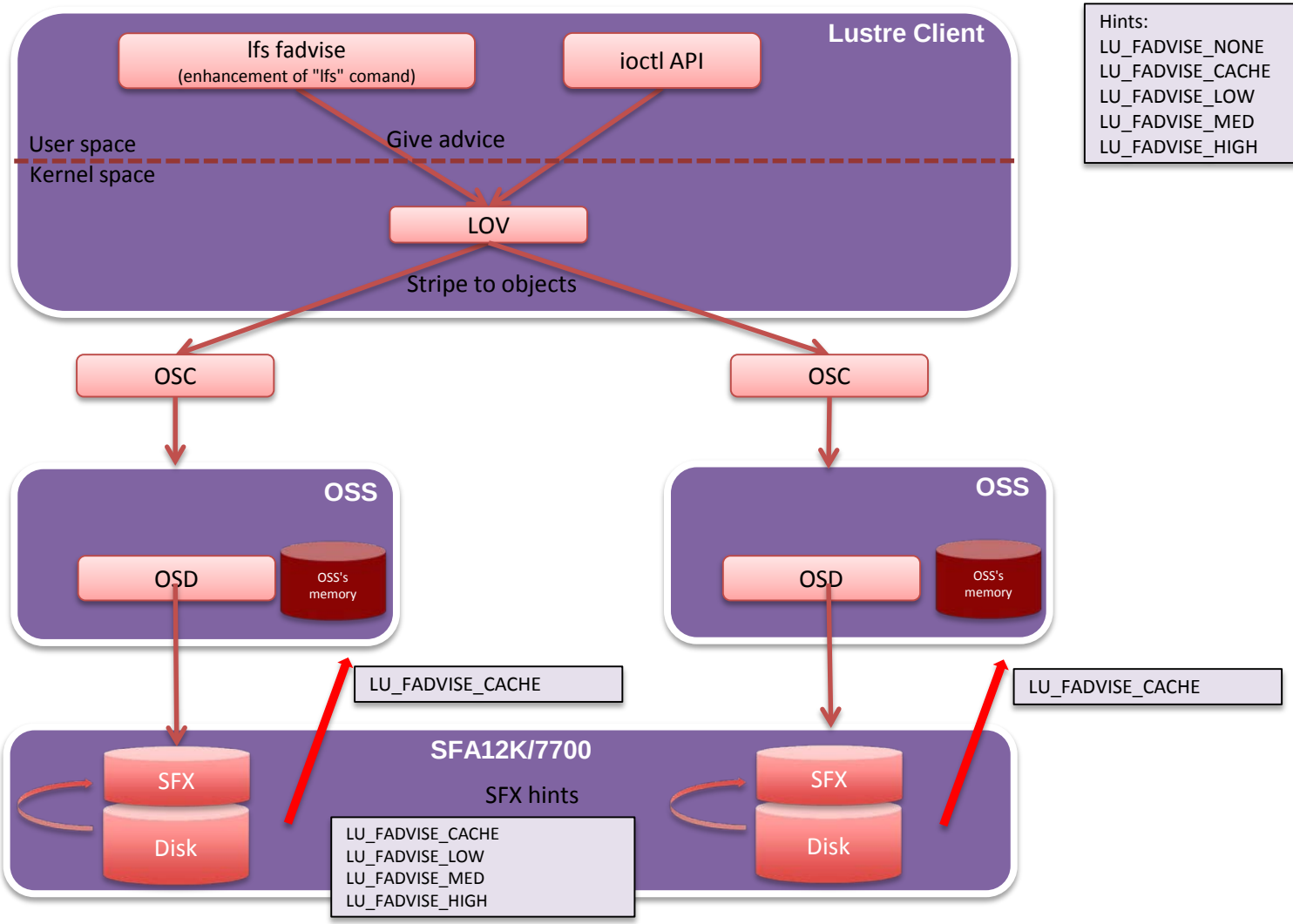
Overview

- The problem: Much interest in broadening applicability of Parallel file systems to cope with small files and small IO
- Small files gives rise to issues in platter usage efficiency, metadata performance and small io performance
- Read cacheing is one aspect but needs added intelligence to preload data

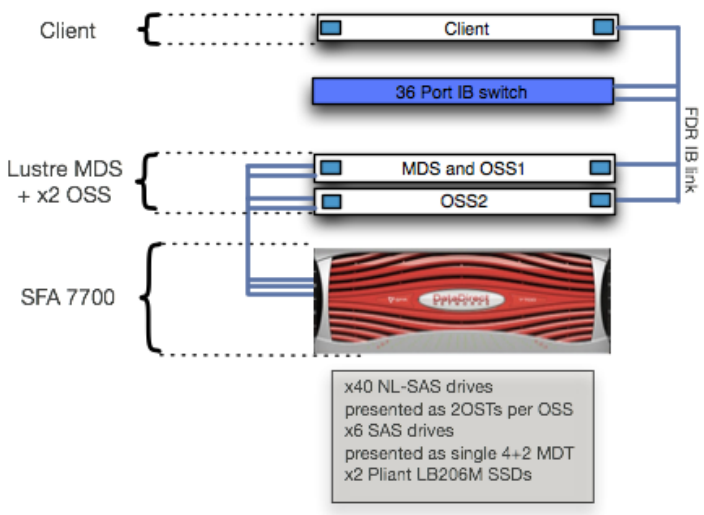
New: Lustre fadvise() framework

- A simple enhancement of Lustre to enable the use higher levels of global cache for reads via fadvise()
- Integrated with Lustre stack
  - Transparently handle file stripe of Lustre
  - Give hints through the I/O path to keep efficient
- Utility and API is simple to use
  - “lfs fadvise” command to give advice on Lustre files
  - ioctl(LL_IOC_FADVISE) for advices from smart applications
- Enable users to give priority hints to SFX hardware from Lustre clients efficiently
- Extensible functionality..
  - Memory based cache on OST side for performance boost
  - In-band priority hints for SFX
  - More types of advice could be implemented based on a common framework
- Possible to be merged in mainline of Lustre codes except SFX codes

System Architecture



Benchmark Setup



Simple Lustre Setup

- Single SFA7700 for OSTs and MDTs
- incorporates 2 SSDs

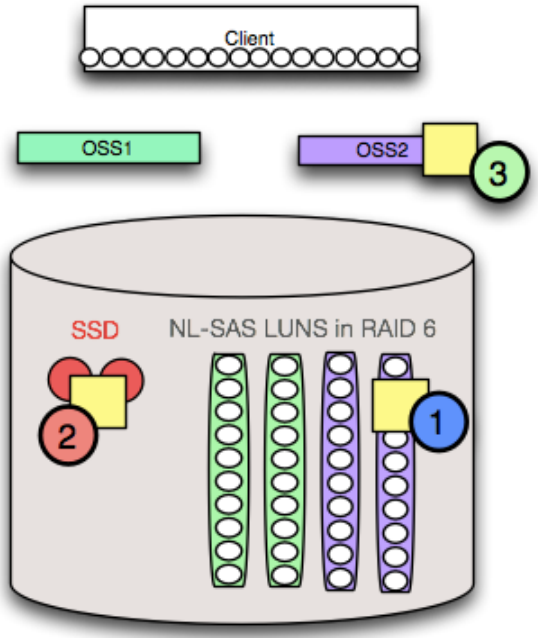
Benchmark

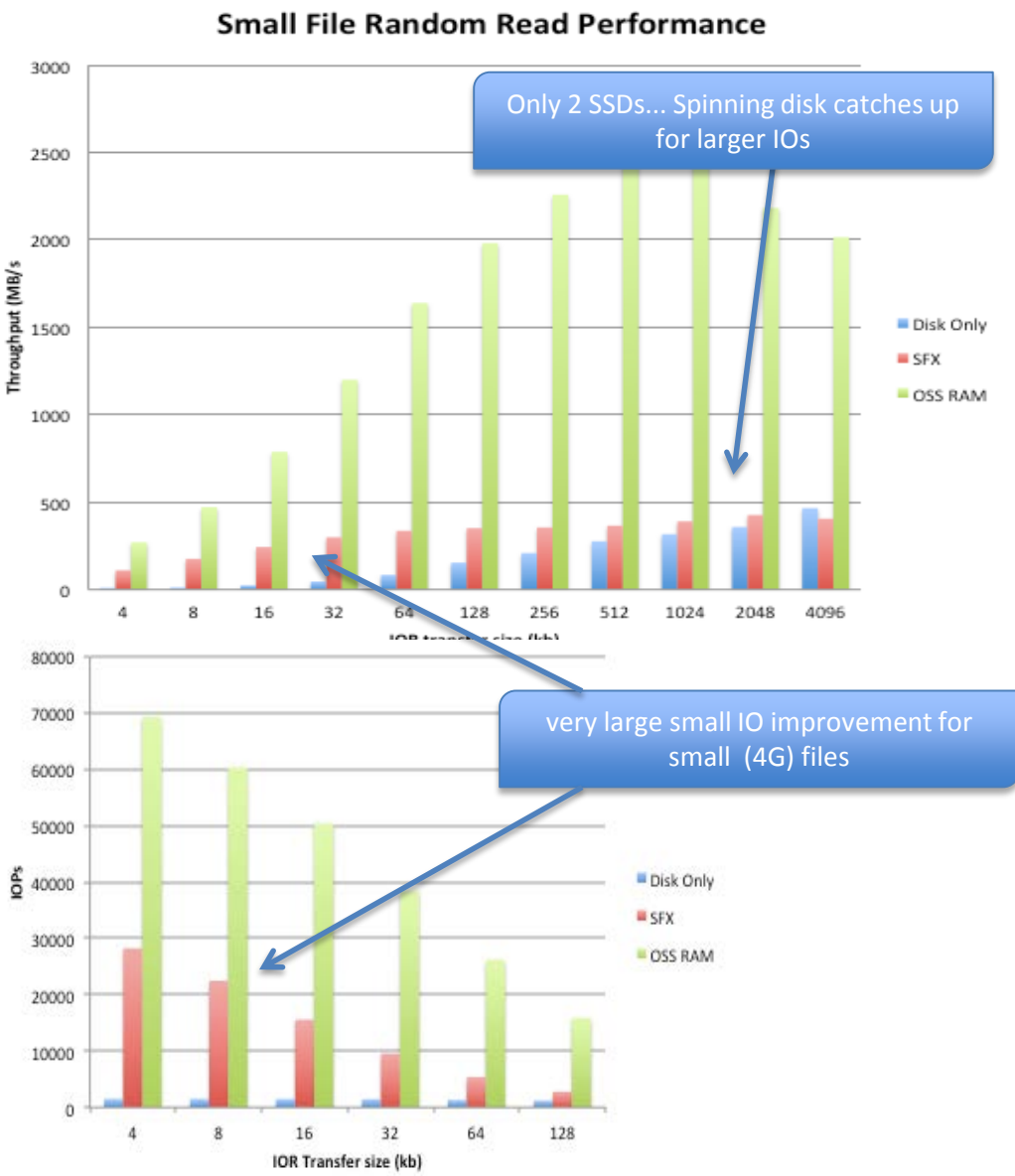
- mpirun -np 16 IOR -r -t 4k -b 1g -e -k -z -o /lustre/file
- One or more large files created. Each one of 16 task reads 4k randomly across the file
  - -b N blockSize -- contiguous bytes to write per task (e.g.: 8, 4k, 2m, 1g)
  - -e fsync -- perform fsync upon POSIX write close
  - -k keepFile -- don't remove the test file(s) on program exit
  - -r readFile -- read existing file
  - -t N transferSize -- size of transfer in bytes (e.g.: 8, 4k, 2m, 1g)
  - -z randomOffset -- access is to random, not sequential, offsets within a file
- Use lfs fadvise and vary argument:
  - lfs fadvise -a high <filename>
  - lfs fadvise <filename>
- Clear client caches between runs...

Benchmark Results

Small Shared File Read

- Create a 4G file and randomly read from many threads on one client using
  1. No Acceleration
  2. Prefetch data to controller SSDs (SFX)
  3. Prefetch data to OSS RAM
- Non-Accelerated file reads limited by the spindles on the OST (8+2)

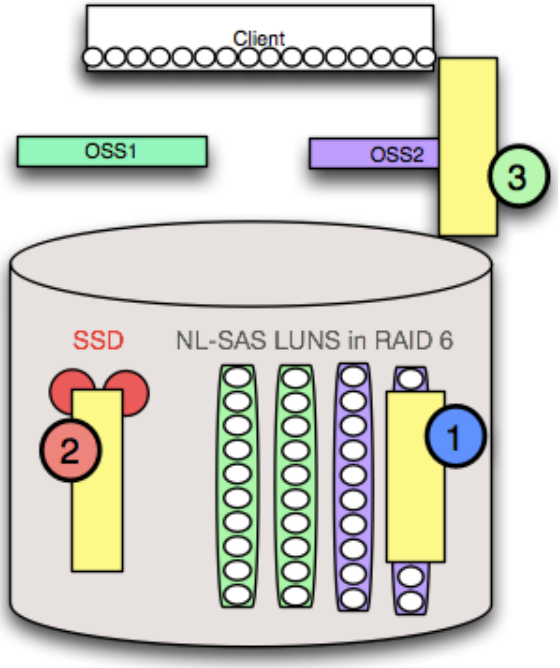


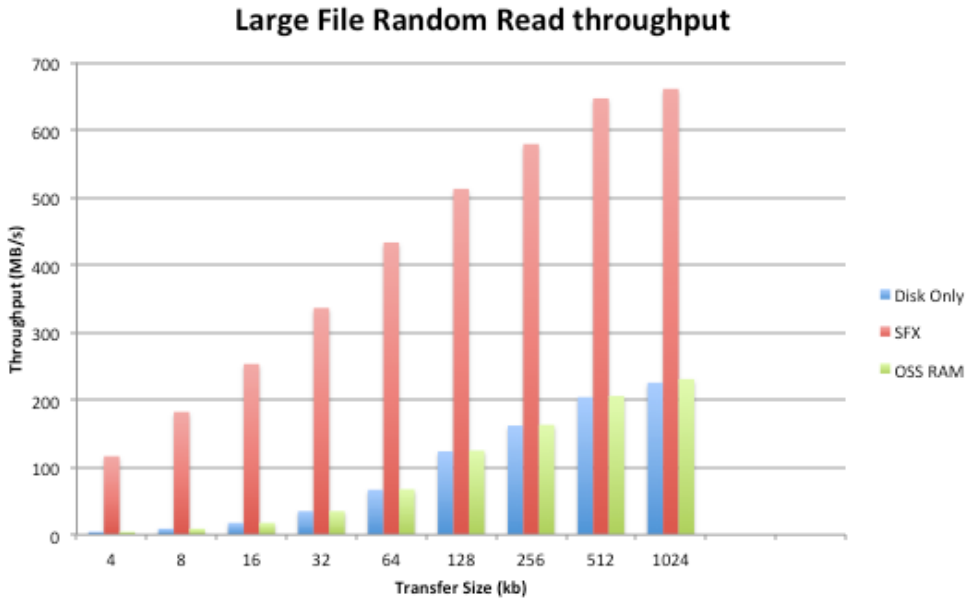


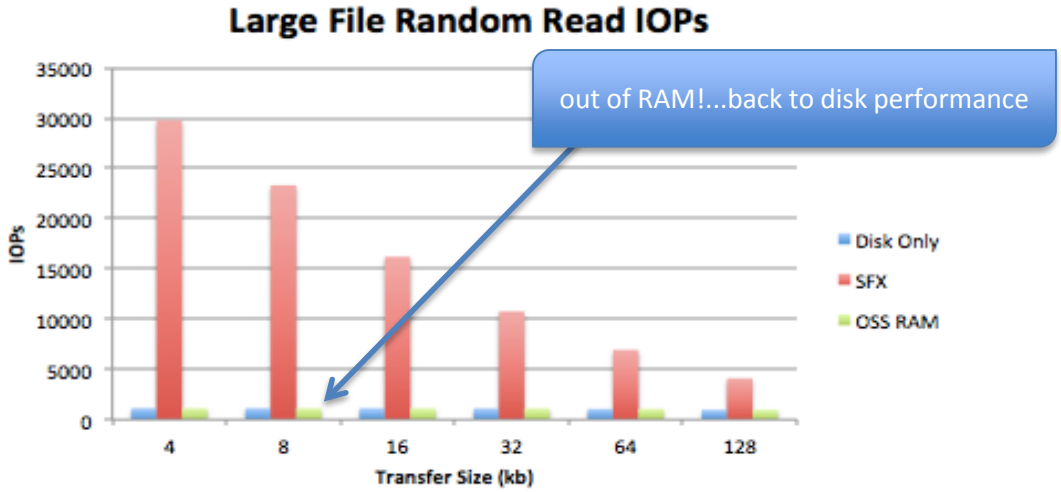
Benchmark Results

Large Shared File Read

- Create a 128G file and randomly read from many threads on one client using
  1. No Acceleration
  2. Prefetch data to controller SSDs (SFX)
  3. Prefetch data to OSS RAM



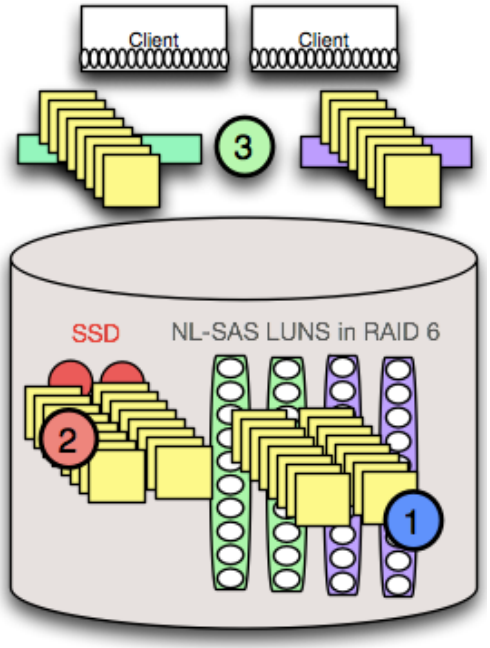


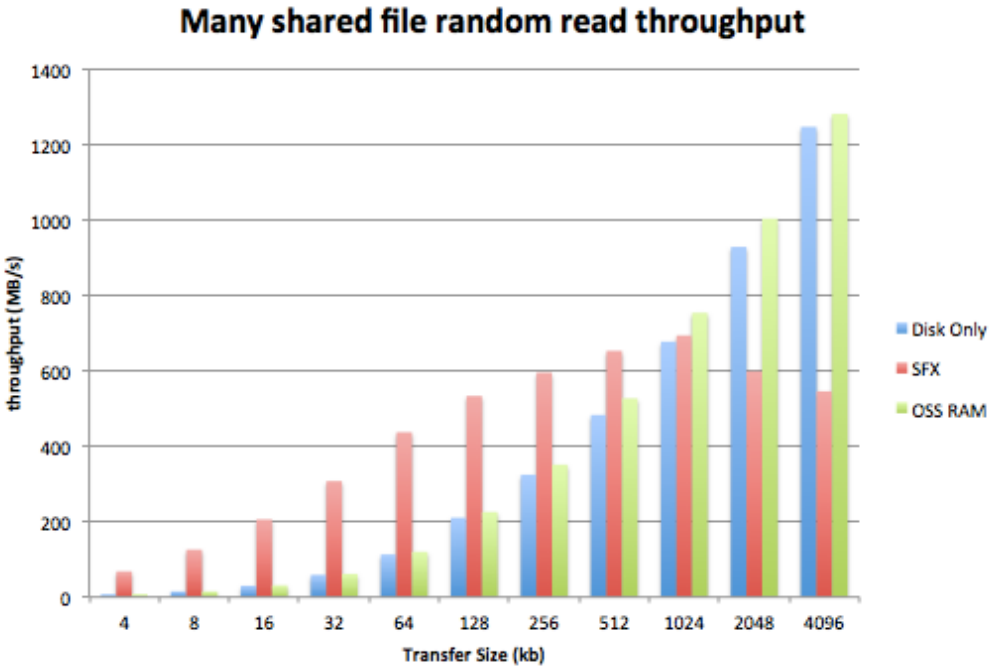


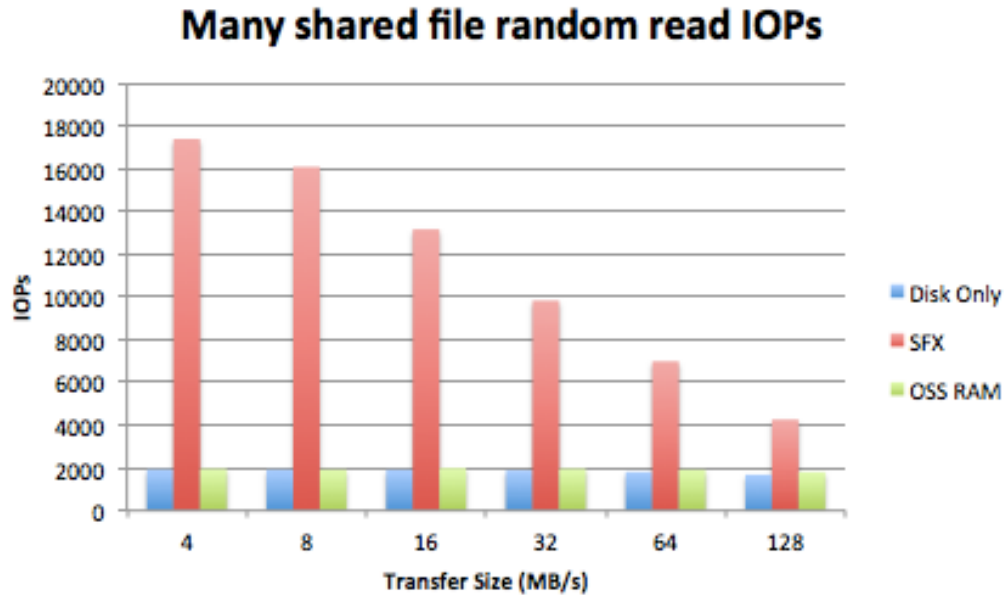
Benchmark Results

Many Files Shared Read

- Create 16x 8G file with Stripe Count=1 across all OSTs and randomly read from many threads on two clients using
  1. No Acceleration
  2. Prefetch data to controller SSDs (SFX)
  3. Prefetch data to OSS RAM







Conclusions

1. Lustre FADVISE() is another useful performance enhancement for small IO reads and can help larger IO too depending on the storage system specifications
2. Can be use for smaller datasets to take advantage of the OSS RAM
3. larger datasets can use an SSD layer such as with SFX
4. Integration with common schedulers possible
5. testing integration into real world environments underway – assistance welcome