A 12 Step Program for Lustre Filesystem Addiction

Shawn Hall and Kent Blancett

BP's HPC environment

- Vital to advancing BP's seismic imaging capabilities
- 3.8 PF of computing power
- 1.3 PB of memory

Courtesy: BP (https://flic.kr/p/gT3aNU)

BP's Lustre environment

- 5300 Lustre clients (no lnet routers)
- All compute systems see all storage
- 20 PB across 4 Lustre file systems (DDN storage)
- Lustre 2.5.27 on servers
- Mix of Lustre 2.3.60, 2.4.1 + patches, 2.5.58 on clients
- 40 Gigabit Ethernet Arista core network with 40 GbE to Lustre servers and 10 GbE from top of rack to compute nodes
- No systematic backups and no purge policy

I/O patterns

- Majority of work uses key-value style data, keys in small file, values in large file
- Often 1 or M → N input and N → N or M output
- Most applications are written in-house
- Like everyone else we have the typical set of degenerate workloads

The 12 Steps

1. Designing

- When designing a Lustre system, need to balance all components
- Think about data flow from disk to client – are you getting the maximum performance from your disks?
- Don’t forget about caching – e.g. do you have enough network bandwidth to make use of the Lustre read cache?
- Do you have single points of failure?

2. Preparing

- Will users require NFS/CIFS access to Lustre?
  - If so, be prepared for instability
  - Firewalls come in handy for blocking users who abuse this service
- What should the default stripe count be?
  - For us, $default\ stripe\ count = \left\lceil \frac{\max\ client\ bandwidth}{\max\ OST\ bandwidth} \right\rceil$
  - Too small and users run into file size limits
  - Too small and 1→N reads limited
  - Too large and increased file system contention

3. Data protection

- Most of us can't back up our entire file systems
- Most users don't think about the safety of their data
- What do we do at BP?
  - Provide straightforward facilities for users to back up their own data and assistance to anyone interested
  - Help identify important files for users
  - Capture full file listing of filesystems nightly
    - Including inode numbers
    - Proved incredibly useful in last year's filesystem crash

4. Education - user

- Teach users the value of backups
- As you get to know the user, you'll know how much information to give them (and what they don't care to know)
- Provide general rules of thumb for I/O
  - Set low default stripe count but allow users to increase
  - Plan ahead on stripe count (e.g. wider stripe for 1→N read files)
  - Don't put 100,000+ files in one directory
  - Avoid putting log files on Lustre, especially when 2000 nodes are simultaneously appending to a single log file

5. Education - administrator

- Storage systems are a critical part of the infrastructure, so they should be an equally important part of an admin's education
- We learn from the community – especially the leaders that are doing innovative work
- Try to leverage the tools others have already written
- Lustre administration isn't passive – Lustre is best managed with active learning and improvement

6. Monitoring

- Lustre Monitoring Tool – how is the filesystem performing?

```
Filesystem: lc1                                                Tue Oct  5 09:03:53 2010
    Inodes:   446.432m total,    52.729m used ( 12%),    393.703m free
     Space:   172.188t total,   138.933t used ( 81%),     33.255t free
   Bytes/s:     0.000g read,       0.294g write,                337 IOPS
   MDops/s:     314 open,       156 close,     533 getattr,       6 setattr
                  4 link,       196 unlink,    434 mkdir,       335 rmdir
                  1 statfs,       3 rename,      0 getxattr
```

>OST	S	OSS	Exp	CR	rMB/s	wMB/s	IOPS	LOCKS	LGR	LCR	%cpu	%mem	%spc
0000	F	tycho1	148	0	0	0	0	382	5	8	1	99	82
0001	F	tycho2	148	0	0	0	1	431	12	23	6	99	81
0002	F	tycho3	148	0	0	1	1	430	0	0	1	84	81
0003	F	tycho4	148	0	0	0	1	855	8	14	3	99	81
0004	F	tycho5	148	0	0	12	12	428	0	0	5	99	82
0005	F	tycho6	148	0	0	9	9	478	6	9	2	82	81
0006	F	tycho7	148	0	0	0	1	369	2	4	5	49	82
0007	F	tycho8	148	0	0	0	1	398	4	9	0	99	81
0008	F	tycho1	148	0	0	0	1	417	3	5	1	99	81
0009	F	tycho2	148	0	0	1	1	415	8	11	6	99	81
000a	F	tycho3	148	0	0	1	2	425	0	0	1	84	81
000b	F	tycho4	148	0	0	12	12	421	5	8	3	99	82
000c	F	tycho5	148	0	0	1	1	446	0	0	5	99	80

Courtesy: LLNL, http://cdn.opensfs.org/wp-content/uploads/2012/12/400-430_Chris_Morrone_LMT_v2.pdf

7. Monitoring

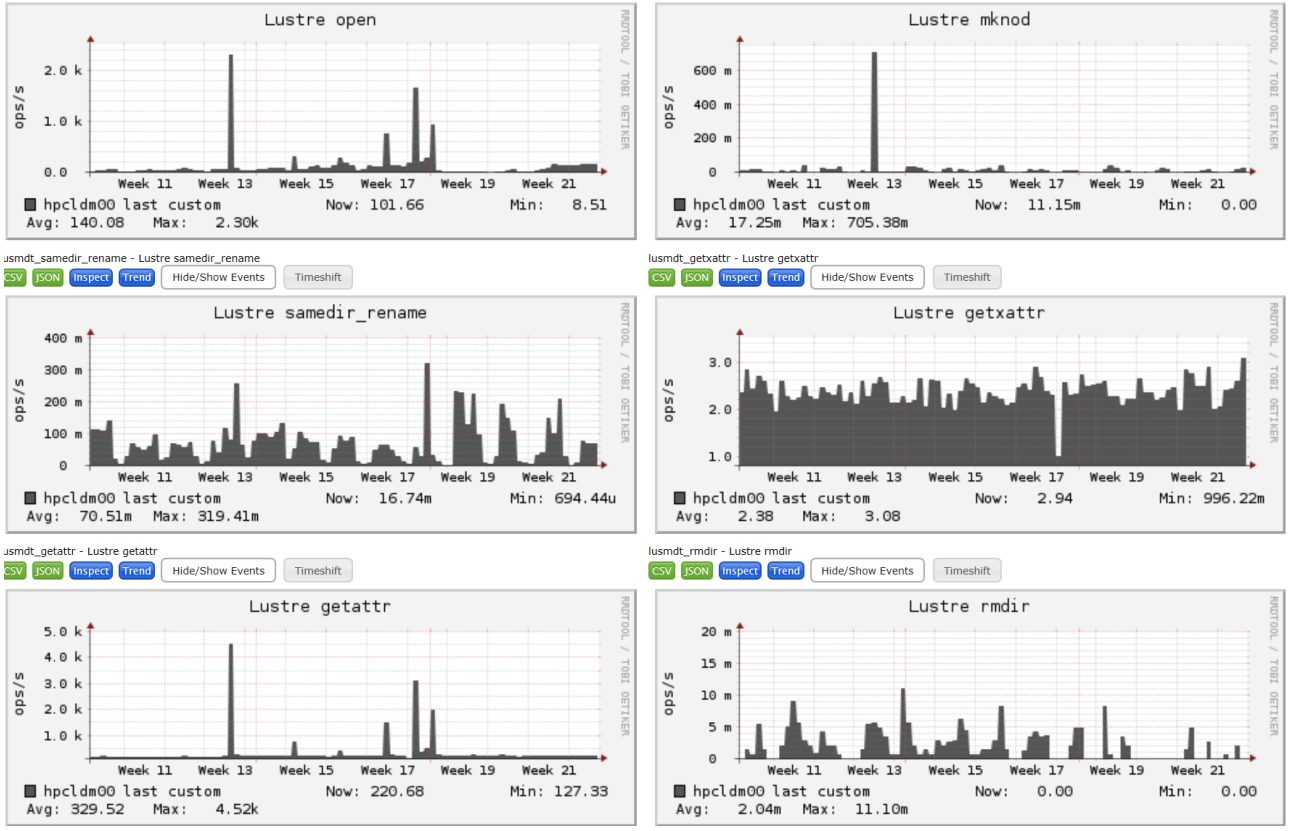
- xltop – who is using the file system?
- Looking into how jobstats can be useful to us

```
FILESYSTEM       MDS/T  LOAD1  LOAD5 LOAD15  TASKS    OSS/T  LOAD1  LOAD5 LOAD15  TASKS  NIDS
ranger-work        1/1   1.52   3.48   4.41    609    14/84   2.74   2.08   2.09   1347  4212
ranger-scratch     1/1   0.13   0.20   0.54    584   50/300   2.52   1.94   1.52   1348  4213
ranger-share       1/1   0.93   1.20   1.72    544     6/36   3.55   1.37   0.90   1203  3960
JOB      FS                 WR_MB/S     RD_MB/S      REQS/S  OWNER     NAME         HOSTS
2526717  ranger-scratch     321.557       5.994    3556.133  tg803155  NST3.28-r0      20
login4   ranger-scratch      38.489      55.054     469.943  NONE      NONE             1
2530927  ranger-scratch      16.526       0.000      39.942  dkcira    Parametric       1
2529449  ranger-work         11.754       0.000      24.088  bealing   PE-OH            4
2530975  ranger-work         11.108       0.007      23.620  vishnam2  batch           16
```

Courtesy: TACC, https://wiki.hpdd.intel.com/display/PUB/Third+Party+Tools

8. Monitoring

- Ganglia – what was the past performance?



9. Bag of Tricks

- Sometimes users have hundreds or thousands of jobs that read the same input file, which hammers the OST (usually mostly hits the cache)
- We occasionally fix this as the job is running if the file being read is small enough
  - Use ltop to find what OSTs are getting hit
  - Use xltop to make an educated guess at what job(s) are causing the load
  - Go to a compute node in the job and use `lsof` to find the potential files
  - Determine the file by using `lfs getstripe` to check the striping of each file and compare to the OSTs that are getting hit
  - Run `lfs migrate --block -c <new stripe count> <file>`
    - This does a copy and some magic to change the layout of an in-use (reads, not writes) file (LU-2445)

10. Bag of Tricks

- Especially on filesystems with no purge policies, they can easily fill up
- We try to help this by
  - Adjust the threshold at which Lustre switches from the round robin to the weighted allocator `lctl set_param lov.lustre?-MDT0000-mdtlov.qos_threshold_rr=??`
  - Adjust the weighting between using free space and location with `lctl conf_param lustre?-MDT0000.lov.qos_prio_free=??`
  - Do whatever we can to shut off object creation (LU-4825)
  - Find large files on full OSTs with `lfs find /<fsname> -obd <OSTname> -size +100G -print`
  - Restripe large files with `/usr/bin/lfs_migrate -y -c <stripe_count> <filename>`

11. Bag of Tricks

- Sometimes Lustre servers will have high load for seemingly no reason
- Use xltop to start looking for suspicious jobs
- See if jobs are getting CPU time
- Strace to see if pauses are on reads and writes

12. Lustre as a Site Wide Filesystem

- Finding combination of supported client and server versions a concern
  - We have clients as old as RHEL 5 and clients coming soon using SLES 12 and/or RHEL 7
- Having a mixed speed Ethernet or Infiniband network also presents challenges
  - After identifying the problem, we had to propogate proper flow control rules across our entire 10/40 GbE network
- Mixed batch and interactive use can be painful for interactive users

Bonus: Our wishlist

- Try out ZFS based Lustre
- Update health scripts for new file systems and make more intelligent
- Enable Lustre jobstats and store them (OpenTSDB?) with graphing capabilities (Grafana?)
- Set up Robinhood instance for each file system

Questions?