Lustre tools for ldiskfs investigation and lightweight I/O statistics

Roland Laifer
Overview

- Lustre systems at KIT
  - Short preview on our next Lustre system
- Lessons learned from wrong quota investigation
  - Developed tools for ldiskfs investigation
- How to easily provide I/O statistics to users
  - Developed tools for lightweight Lustre jobstats usage
Lustre systems at KIT - diagram

- One InfiniBand fabric
- Easy flexible mount on all clusters
- Same Lustre home directory on all clusters

Existing systems at Campus South

- Cluster 1
- Cluster 2
- Cluster 3
- Cluster 4

New systems at Campus North

- ForHLR II Cluster
- ForHLR II Visualization
- Home 1
- Work 1
- Work 2

InfiniBand

- 320 Gbit/s
- 35 km

Lustre 1-4

Lustre 5-7

Visualization

320 Gbit/s

35 km
Lustre systems at KIT - details

<table>
<thead>
<tr>
<th></th>
<th>pfs2</th>
<th>pfs3</th>
<th>pfs4 (Dec 15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>System name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Users</td>
<td>universities, 4 clusters</td>
<td>universities, tier 2 cluster</td>
<td>universities, tier 2 cluster</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(phase 1)</td>
<td>(phase 2)</td>
</tr>
<tr>
<td>Lustre server version</td>
<td>DDN Lustre 2.4.3</td>
<td>DDN based on IEEL 2.2</td>
<td>DDN based on IEEL 2.x</td>
</tr>
<tr>
<td># of clients</td>
<td>1941</td>
<td>540</td>
<td>1200</td>
</tr>
<tr>
<td># of servers</td>
<td>21</td>
<td>17</td>
<td>23</td>
</tr>
<tr>
<td># of file systems</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td># of OSTs</td>
<td>2<em>20, 2</em>40</td>
<td>1<em>20, 2</em>40</td>
<td>1<em>14, 1</em>28, 1*70</td>
</tr>
<tr>
<td>Capacity (TB)</td>
<td>2<em>427, 2</em>853</td>
<td>1<em>427, 2</em>853</td>
<td>1<em>610, 1</em>1220, 1*3050</td>
</tr>
<tr>
<td>Throughput (GB/s)</td>
<td>2<em>8, 2</em>16</td>
<td>1<em>8, 2</em>16</td>
<td>1<em>10, 1</em>20, 1*50</td>
</tr>
<tr>
<td>Storage hardware</td>
<td>DDN SFA12K</td>
<td>DDN SFA12K</td>
<td>DDN ES7K</td>
</tr>
<tr>
<td># of enclosures</td>
<td>20</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td># of disks</td>
<td>1200</td>
<td>1000</td>
<td>1120</td>
</tr>
</tbody>
</table>
Wrong quota investigation - general

- How we recognized that quotas are wrong
  1. Difference between `du -hs <user dir>` and `lfs quota -u <user> <filesys>`
  2. Perl script sums all user and group quotas per OST
     - Used `/proc/fs/lustre/osd-ldiskfs/<ost>/quota_slave/acct_user & acct_group`
     - Should be very similar but showed few per cent deviation
  3. Perl script walks through file system, sums capacities and compares with quotas
     - User / group capacity quotas were up to 30 % wrong

- Support pointed to LU-4345 ([http://review.whamcloud.com/11435](http://review.whamcloud.com/11435))
  - UID / GID of OST object could be set to random value on ldiskfs
  - Capacity quotas are computed from ldiskfs quotas on OSTs
  - Bug fixed with Lustre 2.5.3 but wrong UID / GID values do not disappear
  - Wrong UID / GID of OST object possibly fixed with LFSCK of Lustre 2.6
Wrong quota investigation - basics

Get parent FID and UID/GID of OST object: debugfs stat
[root@OST0]# statcmd="stat -O/0/$(stat /O/0/$(stat /O/0/d$((71666856 % 32))/71666856" [root@OST0]# debugfs -c -R "$statcmd" /dev/mapper/ost_pfs2wor2_0
User: 8972 Group: 12345 Size: 5
fid: parent=[0x200018a62:0x8a4d:0x0] stripe=0

Get FID for file name: lfs path2fid
[user@client]$ lfs path2fid myfile [0x200018a62:0x8a4d:0x0]

Get OST name for OST index: lctl get_param
[user@client]$ lctl get_param lov.pfs2wor2-*.target_obd 0: pfs2wor2-OST0000_UUID ACTIVE

Get object IDs for file name: lfs getstripe
[user@client]$ lfs getstripe myfile
lmm_stripe_count: 2
obdidx objid objid group
 0 71666856 0x4458ca8 0
 2 72574780 0x453673c 0

Get UID/GID for file name: stat
[user@client]$ stat --format="%u %g" myfile
8972 12345

Get file name for FID: lfs fid2path
[root@client]$ lfs fid2path pfs2wor2
[0x200018a62:0x8a4d:0x0]
<path_to_myfile>/myfile

Get parent FID and UID/GID of OST object: debugfs stat
[root@OST0]# statcmd="stat -O/0/$(stat /O/0/$(stat /O/0/d$((71666856 % 32))/71666856"
User: 8972 Group: 12345 Size: 5
fid: parent=[0x200018a62:0x8a4d:0x0] stripe=0

Get FID for file name: lfs path2fid
[user@client]$ lfs path2fid myfile [0x200018a62:0x8a4d:0x0]

Get OST name for OST index: lctl get_param
[user@client]$ lctl get_param lov.pfs2wor2-*.target_obd 0: pfs2wor2-OST0000_UUID ACTIVE

Get object IDs for file name: lfs getstripe
[user@client]$ lfs getstripe myfile
lmm_stripe_count: 2
obdidx objid objid group
 0 71666856 0x4458ca8 0
 2 72574780 0x453673c 0

Get UID/GID for file name: stat
[user@client]$ stat --format="%u %g" myfile
8972 12345
Wrong quota investigation – details (1)

- **Motivation**
  - Check if bug of LU-4345 caused all quota problems

- **Idea**
  - During production, use debugfs to stat all OST objects
    - compare UID / GID with values on file system (MDS)
  - Get biggest object ID:
    - `debugfs -c -R "dump /O/0/LAST_ID /tmp/LAST_ID" <OST device>`
    - `od -Ax -td8 /tmp/LAST_ID`
  - Show object status on ldiskfs:
    - `debugfs -c -R "stat /O/0/d$<object ID modulo 32>/<object ID>" <OST device>`

- **Problem**
  - How to do this fast enough?
Wrong quota investigation – details (2)

Solution

- Perl script pipes many stat commands to same debugfs call
  - Number of commands and when to restart debugfs configurable
- Use another perl script to filter output
  - If object belongs to inspected user print object ID and parent FID
  - OR: Check if UID and GID is inside valid range

Results

- Investigated only one OST
  - Investigation is still time consuming i.e. can take days
- Indeed found lots of objects with wrong UID / GID
- Found a number of orphaned OST objects
  - Unknown reason why they still existed
- Used procedure also helpful for other investigations
Lightweight I/O statistics - diagram

1. request I/O statistics during job submit

2. start batch job

3. send I/O usage of job with Lustre jobstats

4. after job completion create file to request I/O statistics

5. collect new I/O statistics request files

6. collect jobstats from all servers

7. when requested send I/O statistics of job via email

user

batch system

file server

client
Lightweight I/O statistics – steps in detail (1)

1) Enable jobstats for all file systems
   - on clients: `lctl set_param jobid_var=SLURM_JOB_ID`
   - Make sure clients have fix of LU-5179
   - Slurm job IDs are used by Lustre to collect I/O stats
   - On servers increase time for holding jobstats
     - E.g. to 1 hour: `lctl set_param *.*.job_cleanup_interval=3600`

2) User requests I/O statistics with Moab `msub` options:
   - `-W lustrestats:<file system name>[,<file system name>]`
   - Optionally: `-M <email address>`

3) On job completion Moab creates files to request I/O stats
   - File name: `lustrestat-<file system name>-<cluster name>-<job ID>`
   - File content: account name and optionally email address
Lightweight I/O statistics – steps in detail (2)

4) Perl script runs hourly on each file system
   - Uses different config file for each file system
     - Defines names of request files and of batch system servers
       - Allows to collect request files from different clusters
     - Defines which servers are used for the file system
   - Transfers files from batch systems and deletes remote files
     - Uses rsync and rrsync as restricted ssh command for login with key
   - Reads data including job IDs and account name
     - If not specified asks directory service to get email address of account
   - Collects and summarizes jobstats from all servers
   - For each job sends an email
     - Email is good since jobstats are collected asynchronously
Lightweight I/O statistics – example email

Subject: Lustre stats of your job 1141 on cluster xyz

Hello,

this is the Lustre IO statistics as requested by user john_doe on cluster xyz for file system home.

Job 1141 has done ...

... 1 open operations.
... 1 close operations.
... 1 punch operations.
... 1 setattr operations.
... 10 write operations and sum of 10,485,760 byte writes (min IO size: 1048576, max IO size: 1048576).
Lightweight I/O statistics – experiences

- Users do not care much about their I/O usage
  - Tool was not yet frequently used
- No negative impact of jobstats activation
  - Running since 6 weeks
- Another perl script checks high I/O usage per job
  - Collects and summarizes jobstats from all servers
  - Reports job IDs over high water mark for read/write or metadata operations
  - Extremely useful to identify bad file system usage
Summary

- Currently our main Lustre problems are related to quotas
  - Tools helped to analyze on the ldiskfs level
  - New LFSCK features will hopefully fix wrong quotas
  - Quotas on directory tree would be very helpful

- Lustre jobstats are extremely useful
  - Not available with other file systems
  - It’s incredible what users are doing

- All my talks about Lustre

- roland.laifer@kit.edu