Lustre Performance Analysis with SystemTap

Jason Rappleye

NASA Advanced Supercomputing Division

April 24, 2012
Introduction

- Need actionable performance and troubleshooting data at interactive speeds.
- /proc and lctl dk are useful, but:
  - Performance issues at scale
  - Don’t want to pollute logs
  - Want more information
SystemTap

- SystemTap consists of a scripting language, translator and runtime.
- Provides system-wide tracing capabilities.
  - Kernel and userspace.
- `strace` traces a process tree; SystemTap provides visibility across the entire system.
Using SystemTap with Lustre

- Where to probe?
  - This is the hard part - need to have some understanding of how Lustre works.
- Extract data from functions as they are called/return.
  - Output as you go, or
  - Aggregate and periodically display
- Timing function calls.
  - Lustre service threads handle RPCs from start to finish, one at a time
  - Makes it easy to store timing, other information based on the thread handling the request.
Example: Timing ldiskfs block allocations

global start
global times
probe
   module("ldiskfs").function("ldiskfs_mb_new_blocks") {
      start[tid()] = gettimeofday_ms();
   }
probe
   module("ldiskfs").function("ldiskfs_mb_new_blocks").return {
      if ([tid()] in start) {
         times <<= gettimeofday_ms() - start[tid()];
      }
   }
probe end {
   print(@hist_log(times));
}
## Output

<table>
<thead>
<tr>
<th>value</th>
<th>--------------</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>@@@@@@@@@@@@@@</td>
<td>27083</td>
</tr>
<tr>
<td>1</td>
<td>@</td>
<td>675</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>158</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>74</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>128</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>256</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>512</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>~</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 32768 |               | 1     | <---- That’s between 32 and 64 seconds!
Examples

- Poor choice of stripe count.
- Fragmentation.
- High OSS load average.
Poor choice of stripe count

- The default stripe count on our filesystems is one.
- Average file size is relatively small, so this is OK.
- Except...
  - Large tar files can take up a significant portion of an OST.
  - Many ranks writing to a file on one OST can perform poorly.
A SystemTap script intercepts calls in the write path on the OSS to gather the following information:
- NID
- OST name
- object ID
- FID
- UID
- object size

When there’s a write to an object over a predetermined size, print it.

A Python wrapper gathers additional information about the object and writing process, including the path.
Fri Mar 23 10:21:09 2012 service61-ib1 ost:nbp2-OST0021
stripes:1 pid:4320 command:tar size:506123MB
name:/nobackuppp2/.../something.tar
Fragmentation

- **On-disk**
  - Block allocator can’t find a large enough chunk of contiguous free space.
  - Slows down writes; fragmented allocation will cause more I/Os for both reads and writes.

- **Memory**
  - The IB SRP driver can only handle scatter-gather descriptors up to length 255.
Showing I/O fragmentation in real-time

- Use SystemTap to hook into Lustre I/O path.
- A good I/O - 1MB or more in a single write:

  nid:10.151.18.95@o2ib0 ost:nbp2-OST0010 uid:0
  mdt_inode:0 sizes:256

- Memory fragmentation causing SRP to issue multiple I/Os:

  nid:10.151.14.211@o2ib0 ost:nbp2-OST0008 uid:0 mdt_inode:0
  sizes:255(255) 1
On-disk fragmentation

nid:10.151.32.127@o2ib0 ost:nbp2-OST0068 uid:12137
mdt_inode:200443860 sizes:6(6) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1(1) 1
High OSS load average

- High OSS load average is often due to long disk queues.
- A typical cause is many hosts performing I/O to a file on a small number of OSTs.
- You could mine the data in /proc.
  - On a large system, this takes time.
  - I want to know which file is being accessed.
    - Currently possible for writes.
    - May require modifications to Lustre for reads.
<table>
<thead>
<tr>
<th>OST</th>
<th>r/s</th>
<th>w/s</th>
<th>aveq</th>
<th>rwat</th>
<th>wwat</th>
<th>%u</th>
<th>job/host</th>
<th>Ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>nbp2-OST06</td>
<td>80</td>
<td>42</td>
<td>4</td>
<td>48</td>
<td>0</td>
<td>72</td>
<td>23925.pbspl3</td>
<td>8</td>
</tr>
<tr>
<td>nbp2-OST0e</td>
<td>37</td>
<td>2</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>19</td>
<td>66274.pbspl1</td>
<td>6</td>
</tr>
<tr>
<td>nbp2-OST16</td>
<td>50</td>
<td>0</td>
<td>1</td>
<td>18</td>
<td>3</td>
<td>27</td>
<td>66348.pbspl1</td>
<td>6</td>
</tr>
<tr>
<td>nbp2-OST1e</td>
<td>24</td>
<td>0</td>
<td>1</td>
<td>55</td>
<td>1</td>
<td>37</td>
<td>66273.pbspl1</td>
<td>8</td>
</tr>
<tr>
<td>nbp2-OST26</td>
<td>44</td>
<td>18</td>
<td>3</td>
<td>82</td>
<td>0</td>
<td>59</td>
<td>66283.pbspl1</td>
<td>6</td>
</tr>
<tr>
<td>nbp2-OST2e</td>
<td>79</td>
<td>9</td>
<td>1</td>
<td>21</td>
<td>0</td>
<td>39</td>
<td>66428.pbspl1</td>
<td>6</td>
</tr>
<tr>
<td>nbp2-OST36</td>
<td>41</td>
<td>185</td>
<td>135</td>
<td>150</td>
<td>57</td>
<td>100</td>
<td>68894.pbspl1</td>
<td>126</td>
</tr>
<tr>
<td>nbp2-OST3e</td>
<td>80</td>
<td>2</td>
<td>3</td>
<td>34</td>
<td>0</td>
<td>42</td>
<td>68386.pbspl1</td>
<td>12</td>
</tr>
<tr>
<td>nbp2-OST46</td>
<td>63</td>
<td>2</td>
<td>3</td>
<td>48</td>
<td>1</td>
<td>54</td>
<td>66345.pbspl1</td>
<td>6</td>
</tr>
<tr>
<td>nbp2-OST4e</td>
<td>73</td>
<td>2</td>
<td>3</td>
<td>49</td>
<td>0</td>
<td>74</td>
<td>66336.pbspl1</td>
<td>6</td>
</tr>
<tr>
<td>nbp2-OST56</td>
<td>43</td>
<td>13</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>23</td>
<td>66443.pbspl1</td>
<td>6</td>
</tr>
<tr>
<td>nbp2-OST5e</td>
<td>35</td>
<td>1</td>
<td>0</td>
<td>16</td>
<td>6</td>
<td>18</td>
<td>66267.pbspl1</td>
<td>6</td>
</tr>
<tr>
<td>nbp2-OST66</td>
<td>67</td>
<td>1</td>
<td>2</td>
<td>28</td>
<td>1</td>
<td>39</td>
<td>66433.pbspl1</td>
<td>6</td>
</tr>
<tr>
<td>nbp2-OST6e</td>
<td>104</td>
<td>8</td>
<td>3</td>
<td>32</td>
<td>0</td>
<td>49</td>
<td>66278.pbspl1</td>
<td>6</td>
</tr>
</tbody>
</table>
Future work

- Working through the NASA open-source process. Distribution will include:
  - Lustre tapset library
  - big-object and oststat
  - Mechanism for mapping hosts to your site’s batch system
- More tools!
  - Send me your ideas.
  - Better yet, patches :-)
- Visualization
References

- SystemTap
  - http://sourceware.org/systemtap/
- Understanding Lustre Filesystem Internals
Questions?

jason.rappleye@nasa.gov