Analyzing and Predicting LBUG’s with a Hidden Markov Model

Thomas Stibor

t.stibor@gsi.de

High Performance Computing
GSI Helmholtz Centre for Heavy Ion Research
Darmstadt, Germany

Monday 22nd September, 2014

LAD’14 Workshop, Reims, France
Overview

- LBUG’s in Lustre and implications.
- Process Lustre log data with log stash.
- Markov Model (MM).
- Hidden Markov Model (HMM).
- Example (from text processing) what HMM’s can analyze.
- Frequency distributions of function calls.
- Visualized MM transition and HMM emission matrices.
- Sampling and predicting LBUG’s in future time windows.
**LBUG’s and Implications**

- Lustre is a large project with complex code base.
- Lustre is prone to **critical** software bugs called (LBUG’s).
- LBUG is a software behavior that causes freeze of kernel thread and subsequent reboot.

```c
void lbug_with_loc(struct libcfs_debug_msg_data *) __attribute__((noreturn));

#define LBUG() do {
    LIBCFS_DEBUG_MSG_DATA_DECL (msgdata, D_EMERG, NULL);
    lbug_with_loc(&msgdata);
} while(0)
```

```
#define LIBCFS_DEBUG_MSG_DATA_DECL ( dataname , mask , cdls )
static struct libcfs_debug_msg_data dataname = {
    .msg_subsys = DEBUG_SUBSYSTEM,
    .msg_file = __FILE__,
    .msg_fn = __FUNCTION__,
    .msg_line = __LINE__,
    .msg_cdls = (cdls)
};
dataname.msg_mask = (mask);
```

“Note - **LBUG** freezes the thread to allow capture of the panic stack. A system reboot is needed to clear the thread.”

int mdt_getxattr(struct mdt_thread_info *info)
{
    struct ptlrpc_request   *req = mdt_info_req(info);
    struct mdt_export_data  *med = mdt_req2med(req);
    struct lu_ucred          *uc = lu_ucred(info->mti_env);
    ...
    ...
    valid = info->mti_body->valid & (OBD_MD_FLXATTR | OBD_MD_FLXATTRLS);

    if (valid == OBD_MD_FLXATTR) {
        char  *xattr_name = req_capsule_client_get(info->mti_pill, &RMF_NAME);
        rc = mdt_getxattr_one(info, xattr_name, next, buf, med, uc);
    } else if (valid == OBD_MD_FLXATTRLS) {
        CDEBUG(D_INODE, "listxattr \n");
        rc = mo_xattr_list(info->mti_env, next, buf);
        if (rc < 0)
            CDEBUG(D_INFO, "listxattr failed: %d\n", rc);
    } else if (valid == OBD_MD_FLXATTRALL) {
        rc = mdt_getxattr_all(info, reqbody, repbody, buf, next);
    } else
        LBUG();
Lustre Log Data Pre-Processing Steps

1. Fetching entire log data from archive tape.
2. Resulting in “giant” log data.

| drwxr-sr-x 14 root staff 114 Mai 2 11:37 .. |
|-----|------------------|-----------------|-------------|
| -rw-r----- 1 root adm 2,5G Feb 2 2012 syslog -20120201 |
| -rw-r----- 1 root adm 2,0G Feb 3 2012 syslog -20120202 |
| -rw-r----- 1 root adm 2,2G Feb 4 2012 syslog -20120203 |
| -rw-r----- 1 root adm 2,0G Feb 5 2012 syslog -20120204 |
| -rw-r----- 1 root adm 1,9G Feb 6 2012 syslog -20120205 |
| -rw-r----- 1 root adm 1,9G Feb 7 2012 syslog -20120206 |

...)

```
du -sh 2012/02/
57G  02/
```

- Total amount of Lustre log data for 2012 is \(\approx 2.1\) GByte.
- A true **Big Data** problem.
Processed Lustre Logs

Logstash: CSV exported data to be analyzed:

Example 1:

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Node</th>
<th>Type</th>
<th>File</th>
<th>Line</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 26</td>
<td>20:05:28</td>
<td>lxfs290</td>
<td>LustreError</td>
<td>filter.c</td>
<td>2732</td>
<td>__filter_oa2dentry: filter_preprw_read on non-existent object: 10</td>
</tr>
<tr>
<td>Mar 26</td>
<td>20:05:28</td>
<td>lxfs290</td>
<td>LustreError</td>
<td>filter_io.c</td>
<td>488</td>
<td>filter_preprw_read, ASSERTION (PageLocked(lnb-&gt;page)) failed</td>
</tr>
<tr>
<td>Mar 26</td>
<td>20:05:28</td>
<td>lxfs290</td>
<td>LustreError</td>
<td>filter_io.c</td>
<td>488</td>
<td>filter_preprw_read, LBUG</td>
</tr>
</tbody>
</table>

Example 2:

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Node</th>
<th>Type</th>
<th>File</th>
<th>Line</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 27</td>
<td>22:56:01</td>
<td>lxfs124</td>
<td>LustreError</td>
<td>events.c</td>
<td>381</td>
<td>server_bulk_callback, &quot;event type 4, status -5, desc ffff8800c791c000&quot;</td>
</tr>
<tr>
<td>May 28</td>
<td>00:15:50</td>
<td>lxmds11</td>
<td>LustreError</td>
<td>client.c</td>
<td>178</td>
<td>ptrlpc_free_bulk, ASSERTION (atomic_read(&amp;(desc-&gt;bd_export)-&gt;exp_refcount) &lt; 0x5a5a5a) failed</td>
</tr>
<tr>
<td>May 28</td>
<td>00:15:50</td>
<td>lxmds11</td>
<td>LustreError</td>
<td>service.c</td>
<td>1426</td>
<td>ptrlpc_server_handle_request, ASSERTION (atomic_read(&amp;(export)-&gt;exp_refcount) &lt; 0x5a5a5a) failed</td>
</tr>
<tr>
<td>May 28</td>
<td>00:15:50</td>
<td>lxmds11</td>
<td>LustreError</td>
<td>service.c</td>
<td>1426</td>
<td>ptrlpc_server_handle_request, LBUG</td>
</tr>
<tr>
<td>May 28</td>
<td>00:15:50</td>
<td>lxmds11</td>
<td>LustreError</td>
<td>client.c</td>
<td>178</td>
<td>ptrlpc_free_bulk, LBUG</td>
</tr>
</tbody>
</table>

Note: Patch exists for Example 2

---

Lustre / LU-919

Multiple wrong LBUGs checking cfs_atomic_t vars/fields with inaccurate poison value of 0x5a5a5a
Can we predict LBUG’s in Lustre?

Consider this simple approach for modeling Lustre function calls and corresponding LBUG occurrence:

This looks like the familiarized Hidden Markov model.
Markov Model Weather Example

\[ s = \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \]

\[ H = \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix} \]

\[ T \] : Length of observation sequence.

\[ N_Q \] : Number of states in the model.

\[ s \] : Initial state distribution.

\[ H \] : Transition matrix.

\[ \{ Q_1, Q_2, \ldots, Q_T \} \] : Set of time indexed random variables for states.

\[ M = (s, H) \] : Markov model parameters.

Joint distribution for sequence of \( T \) observations
\[ \Pr(Q_1, Q_2, \ldots, Q_T) = \Pr(Q_1) \prod_{t=2}^{T} \Pr(Q_t|Q_1, \ldots, Q_{t-1}) \]

Joint distribution under first-order Markov assumption:
\[ \Pr(Q_1, Q_2, \ldots, Q_T) = \Pr(Q_1) \prod_{t=2}^{T} \Pr(Q_t|Q_{t-1}) \]
Markov Model Weather Example (cont.)

Given that weather on day 1 \( (t = 1) \) is sunny.

- What is the probability for the next 3 days weather will be \( O = \text{“sunny→rainy→rainy”} \)?

\[
\Pr(O|M) = \Pr(Q_1 = \text{sunny}, Q_2 = \text{sunny}, Q_3 = \text{rainy}, Q_4 = \text{rainy}) \\
= \Pr(Q_1 = \text{sunny}) \cdot \Pr(Q_2 = \text{sunny} | Q_1 = \text{sunny}) \\
\cdot \Pr(Q_3 = \text{rainy} | Q_2 = \text{sunny}) \\
\cdot \Pr(Q_4 = \text{rainy} | Q_3 = \text{rainy}) \\
= s_2 \cdot H_{22} \cdot H_{21} \cdot H_{11} \\
= 0.4 \cdot 0.6 \cdot 0.4 \cdot 0.7 = 0.0672
\]

Note: Entries in matrix \( H \) can be interpreted as follows:

\[
H_{ij} = \Pr(Q_{t+1} = j | Q_t = i)
\]

where for the sake of simplicity states are from set \( \{1, 2, \ldots, N_Q\} \)
Hidden Markov models

Suppose we are locked in room without windows, and somebody is telling us the following observations and ask us to tell him what weather is outside:

\[
\mathcal{O} = \text{walk} \rightarrow \text{clean} \rightarrow \text{shop} \rightarrow \cdots \rightarrow \text{shop}
\]

\[
\begin{align*}
\mathbf{s} &= (0.6, 0.4) \\
\mathbf{H} &= \begin{pmatrix}
\text{rainy} & \text{sunny} \\
0.7 & 0.3 \\
0.4 & 0.6
\end{pmatrix} \\
\mathbf{E} &= \begin{pmatrix}
\text{rainy} & \text{walk} & \text{shop} & \text{clean} \\
0.1 & 0.4 & 0.5 \\
0.7 & 0.2 & 0.1
\end{pmatrix}
\end{align*}
\]
Hidden Markov models (cont.)

Problem 1: Calculate probability of observation sequence $O$, given model $M$, that is $\Pr(O \mid M) = ?$

Problem 2: Given HMM and observation sequence $O$, find most likely hidden state sequence.

Problem 3: How do we estimate model parameters $M = (s, H, E)$ to maximize $\Pr(O \mid M)$. Loosely speaking, how do we estimate $M$ that “best” fits our data $O$.

For further details see paper:

Analyze Novel Flatland with a HMM

Example from Sec. 1 Of the Nature of Flatland: by Edwin A. Abbott

I call our world Flatland, not because we call it so, but to make its nature clearer to you, my happy readers, who are privileged to live in Space.

Imagine a vast sheet of paper on which straight Lines, Triangles, Squares, Pentagons, Hexagons, and other figures, ...

Process and convert text into:

Train HMM of following form:
Analyze Novel Flatland with a HMM (cont.)

<table>
<thead>
<tr>
<th>letter</th>
<th>$h_1$</th>
<th>$h_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.00</td>
<td>16.51</td>
</tr>
<tr>
<td>b</td>
<td>2.70</td>
<td>0.00</td>
</tr>
<tr>
<td>c</td>
<td>5.37</td>
<td>0.24</td>
</tr>
<tr>
<td>d</td>
<td>5.06</td>
<td>2.39</td>
</tr>
<tr>
<td>e</td>
<td>0.00</td>
<td>27.10</td>
</tr>
<tr>
<td>f</td>
<td>4.59</td>
<td>0.00</td>
</tr>
<tr>
<td>g</td>
<td>1.84</td>
<td>1.92</td>
</tr>
<tr>
<td>h</td>
<td>10.10</td>
<td>0.00</td>
</tr>
<tr>
<td>i</td>
<td>0.00</td>
<td>16.14</td>
</tr>
<tr>
<td>j</td>
<td>0.16</td>
<td>0.00</td>
</tr>
<tr>
<td>k</td>
<td>0.50</td>
<td>0.16</td>
</tr>
<tr>
<td>l</td>
<td>8.02</td>
<td>0.00</td>
</tr>
<tr>
<td>m</td>
<td>4.97</td>
<td>0.00</td>
</tr>
<tr>
<td>n</td>
<td>13.58</td>
<td>0.00</td>
</tr>
<tr>
<td>o</td>
<td>0.00</td>
<td>17.03</td>
</tr>
<tr>
<td>p</td>
<td>2.55</td>
<td>0.77</td>
</tr>
<tr>
<td>q</td>
<td>0.13</td>
<td>0.13</td>
</tr>
<tr>
<td>r</td>
<td>10.90</td>
<td>0.00</td>
</tr>
<tr>
<td>s</td>
<td>12.06</td>
<td>0.30</td>
</tr>
<tr>
<td>t</td>
<td>9.03</td>
<td>9.29</td>
</tr>
<tr>
<td>u</td>
<td>1.49</td>
<td>4.80</td>
</tr>
<tr>
<td>v</td>
<td>1.78</td>
<td>0.00</td>
</tr>
<tr>
<td>w</td>
<td>3.04</td>
<td>0.34</td>
</tr>
<tr>
<td>x</td>
<td>0.46</td>
<td>0.00</td>
</tr>
<tr>
<td>y</td>
<td>1.46</td>
<td>2.82</td>
</tr>
<tr>
<td>z</td>
<td>0.11</td>
<td>0.00</td>
</tr>
</tbody>
</table>

$\times 100$
Frequency Distribution of Function Calls (1-Gram Seq.)

Ixmds11 function call distribution 01-2013

Ixmds11 function call distribution 02-2013

Ixmds11 function call distribution 03-2013

Ixmds11 function call distribution 04-2013
Visualize Markov Model (2-Gram Sequence) of Fnc. Calls

lxmds11–01–2013.seq
Visualize Markov M. (2-Gram Seq.) of Fnc. Calls (cont.)

lxmds11-02-2013.seq
Visualize Markov M. (2-Gram Seq.) of Fnc. Calls (cont.)

lxmds11-03-2013.seq
Visualize Markov M. (2-Gram Seq.) of Fnc. Calls (cont.)

lxmds11-04-2013.seq
Visualize Markov M. (2-Gram Seq.) of Fnc. Calls (cont.)

lxmds11–05–2013.seq
Visualize Markov M. (2-Gram Seq.) of Fnc. Calls (cont.)

lxmds11-06-2013.seq
Visualize Markov M. (2-Gram Seq.) of Fnc. Calls (cont.)

lxmds11−07−2013.seq
Visualize Markov M. (2-Gram Seq.) of Fnc. Calls (cont.)

lxmds11–08–2013.seq
Visualize HMM of Functions Calls

2 Hidden States
Visualize HMM of Functions Calls (cont.)

3 Hidden States
Visualize HMM of Functions Calls (cont.)

4 Hidden States
LBUG Prediction with HMM

- Train HMM on Lustre logs (function calls, LBUGs)
- Sample function call sequences from HMM.

Example: Sample from trained HMM (2 hidden states, 36 emitting states (function calls))

```
ldlm_handle_enqueue, lov_find_pool, target_handle_connect, target_handle_reconnect, target_handle_reconnect, lov_clear_orphans, ..., target_handle_reconnect, target_handle_reconnect, target_handle_reconnect, LBUG
```

Expected number of LBUG's within the next T time steps, 2 hidden states

![Graph showing expected number of LBUGs over time steps](image)
Summary & Outlook

Analyzing Lustre log files with (Hidden) Markov Models for:

- visualizing and analyzing problems,
- recover latent structures and relations,
- predicting future problems (LBUG’s), by sampling from the model.

HMM implementation (in C) and R scripts (for generating chord diagrams) will be available soon at https://bitbucket.org/tstibor

Many thanks to Matteo Dessalvi for providing Lustre log data and great discussions on machine learning.

Questions?