Online Diagnostics at Scale

Don Maxwell (ORNL)
Jeff Becklehimer (Cray)
April 30, 2012
• Diagnostic definitions
• Diagnostic process
• Motivation and Experiences
• Finding Problems
• Resolving Problems
Diagnostic Definitions

Offline Diagnostics

No operating system booted
Manufacturing screens
Cray

SMW necessary
memtest, cpuburn, mtstat, bist, xtcablecheck

Online Diagnostics

Runs under operating system
Acceptance tests – Batch, Intel MPI Benchmarks (IMB)
Applications that stress the machine
Diagnostic Process

• Commodity Part Manufacturer
  – Screening Process To Eliminate Weak Parts
    • Thermal Extremes
    • Electrical Extremes
    • Application Results

• Supercomputer Vendor
  – Schedules Sometimes Dictate Parts Arrival at Site
  – Screening/Burn-In Process To Eliminate Weak Parts
    • Not as Extreme Extremes
    • Different Set of Applications to Eliminate Weak Parts

• Consumer
  – Screening/Burn-In Process To Eliminate Weak Parts
  – Yet Another Set of Applications
  – Acceptance Process
Motivation

• Why do we care about diagnostics?
  – Reliability
  – Stability
  – Metric requirements
  – User experience

• HPC Pitfalls
  – Volume of parts
  – Early parts
    • Schedules
    • Experience in the market minimal
  – Diagnostic scaling
Experiences

• Early-life failures
  – Node failures
  – Segmentation faults for particular codes
    • LSMS/Madness/S3D
  – Soft errors

• Most problems found during installation and acceptance

• Initial search for ongoing diagnostic to minimize user impact
  – Short-running application proven to find issues
  – Run in nodehealth/NodeKARE?
    • No MPI
  – Prologue/Epilogue
    • Apps not short enough
  – Batch mode
    • Problems
      – Application must scale higher than per node due to ALPS limitations
      – Cycles away from users
        » Can only run a small number of months after acceptance
Finding Soft Errors

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td></td>
</tr>
</tbody>
</table>
MySQL Table for Tracking Diag Runs

- Accounting to cover the entire machine
- Compare (now - last runtime) > run frequency
- Moab reservations used to get jobs through machine
- Experimentation with large jobs vs. small jobs

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>last_runtime</td>
<td>Datetime</td>
</tr>
<tr>
<td>processor_id</td>
<td>int(10) unsigned</td>
</tr>
<tr>
<td>job_id</td>
<td>varchar(80)</td>
</tr>
<tr>
<td>Hostname</td>
<td>varchar(80)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>last_runtime</th>
<th>processor_id</th>
<th>job_id</th>
<th>hostname</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012-04-20 02:23:49</td>
<td>4516</td>
<td>1075439</td>
<td>jaguarpf</td>
</tr>
</tbody>
</table>
Diagnostic Run Flowchart

1. Run Largest Job That Fits
2. Investigate Cause
3. Results Correct?
4. Submit Job against Moab Reservation
5. Investigate Cause
6. Results Correct?
7. Delete Moab Reservation

- Update Database for Verified nodes
- Submit New Large Job to Run at Future Interval
- Query Database for Eligible Nodes (Eliminate Down Nodes)
- Create Moab Reservation based on Eligible Nodes
Segmentation Faults

• Good news
  – Console output to detect problem

• Bad news
  – Cannot distinguish a hardware issue from a user error

A definite hardware problem

[2011-11-15 10:38:48][c8-4c2s0n0]lmp_jaguar_pgi[8292] general protection ip:918202 sp:7fffffff83a0 error:0 in lmp_jaguar_pgi[400000+883000]
[2011-11-16 03:39:08][c8-4c2s0n0]lmp_jaguar_pgi[9425] general protection ip:918202 sp:7fffffff8390 error:0 in lmp_jaguar_pgi[400000+883000]
[2011-11-16 03:49:37][c8-4c2s0n0]lmp_jaguar_pgi[9633] general protection ip:918202 sp:7fffffff8340 error:0 in lmp_jaguar_pgi[400000+883000]

Don't care - software development issue

[2012-04-25 05:48:17][c9-1c2s2n2]namd2-topo-pme[17690] general protection ip:45ffd35 sp:2aab75ef7d90 error:0 in namd2-topo-pme[4000000+a20000]
[2012-04-25 05:48:17][c9-6c1s5n1]namd2-topo-pme[18973] general protection ip:45ffd35 sp:2aab75ef7d90 error:0 in namd2-topo-pme[4000000+a20000]
Segmentation Faults Monitored by SEC

- SEC (Simple Event Correlator)
  - [http://simple-evcorr.sourceforge.net](http://simple-evcorr.sourceforge.net)
  - In use at OLCF since 2006 to monitor systems
    - “Real Time Health Monitoring of the Cray XT Series Using the Simple Event Correlator (SEC),” CUG 2007

```plaintext
Suppress user segfaults based on context

```

```
Set the suppress context when 2 faults from the same app within a given window of time

```

```plaintext

suppress
ptype= RegExp
pattern= \[((\s*[0-9]+)\s*([0-9A-Za-z]+)\]([S+])\[d+\]][:]* (?:general protection|segfault)
context= SUPPRESS_APP_$3

single
ptype= RegExp
pattern= \[((\s*[0-9]+)\s*([0-9A-Za-z]+)\]([S+])\[d+\]][:]* (?:general protection|segfault)
desc= Segfault App $3
action= create SUPPRESS_APP_$3 180; \reset Segfault Node $2 App $3; \delete SEGFAULT_$2_$3
window= 30
thresh= 2
```
Segmentation Faults Monitored by SEC

If not suppressing, add data to context for later reporting if conditions are met. Set expiration in case conditions are not met.

Wait window of time for 3 segfaults from the same app on the same node and report.

type= Single
ptype= RegExp
continue = takenext
pattern= \([(\:\- 0-9]+)\][(0-9A-z-]+)(\S+)\[d+\]][:]* (?:general protection|segfault)
context= !SUPPRESS_APP_$3
desc= Segfault Node $2 App $3
action= add SEGFAULT_$2_$3 $1 Node $2 Segfaulted App $3; \\
        set SEGFAULT_$2_$3 200

Wait window of time for 3 segfaults from the same app on the same node and report.

type= SingleWithThreshold
ptype= RegExp
continue = takenext
pattern= \([(\:\- 0-9]+)\][(0-9A-z-]+)(\S+)\[d+\]][:]* (?:general protection|segfault)
context= SEGFAULT_$2_$3
desc= Segfault Node $2 App $3
action= report SEGFAULT_$2_$3 /bin/mail -s "%%DEST_HOST%% SegFault Threshold Exceeded $2" \\
        %%DEST_EMAIL%%; \\
        delete SEGFAULT_$2_$3
window= 180
thresh= 3
Removing Failed Parts

• Gemini Interconnect
  – Dynamic routing provides capability for replacement of parts on running machine
  – Software bugs cause reroute to fail leading to reboot

• Warmswap Procedure
  1. Verify system reroute
     • rtr –stage-routes
  2. Idle the module
     • Moab hostlist-based reservation
     • Check ALPS to ensure the module is idle
  3. xtwarmswap/Remove/Repair/Replace module
  4. xtbounce/Route/Boot module
  5. Run test jobs against Moab reservation
  6. If successful, remove Moab reservation – otherwise, return to hardware engineers
Future

- Reduce application to run on the order of seconds
- Ensure application can run in a scalable manner
- Integrate margining extremes to verify parts
- Incorporate into prologue/epilogue