A State Machine Approach to Disambiguating Supercomputer Event Logs

Jon Stearley, Robert Ballance, Lara Bauman

USENIX Workshop on Managing Systems Automatically and Dynamically (MAD’12)

Oct 7, 2012
For accurate understanding of our systems, event disambiguation is essential!

Ambiguous - all events of interest

One dot per event
(708,510 total)

Which of these dots indicate “failures”?

How would you calculate mean time to failure?

Or model failure distribution, etc?

These are only the events that system administrators or research papers have identified as significant from a reliability perspective.
For accurate understanding of our systems, event disambiguation is essential!

One dot per event
(70,126 total)

Which of these dots indicate “failures”?

How would you calculate mean time to failure?

Or model failure distribution, etc?

Do these lines indicate scheduled reboots?
*(only the system administrators know...)*

Are these partial lines benign reboots, or failures?
Disambiguated - CRAYHEARTSTOPALERT

One dot per event
(70,126 total)

Which of these dots indicate “failures”?
(1,784 total, at 77 distinct times)

How would you calculate mean time to failure?

Or model failure distribution, etc?

For accurate understanding of our systems, event disambiguation is essential!
**Event Classes**

- **All Messages**
  - **Messages of Interest**
  - **Possible State-Changes**
    - **Actual State-Changes**

**Example Events**

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Splunk syntax</th>
<th>Example Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRAYHEARTSTOPALERT</td>
<td>sourcetype=cray ec_heartbeat_stop &quot;considered dead&quot;</td>
<td>2012-05-29 12:55:45 ec_heartbeat_stop src::c13-lc2s1 seqnum:0x0 svc::c13-lc2s1n1(131276)[alert] node heartbeat fault] Cause:ec_null Text:node c13-lc2s1n1 considered dead, last heartbeat 00025e67, HT cave indicates a HT link is down, HT lookup criteria met, HT lookup reset is on, socket 0 core 0 bank 0 status 0xb6000000000000185 addr 0x00000000000000400 ...</td>
</tr>
<tr>
<td>MOABRSVSTARDSYS</td>
<td>sourcetype=moabstats etype=rsvstart rsvid=PreventMaint.* OR rsvid=system* OR rsvid=<em>-sys.</em></td>
<td>05:24:50 1338290685:2551224 rsv PreventMaint.2 RSVSTART 1338290660 1338290660 1538290660 8894 8894 0.000000 0.000000 92,93,2,3,9212,...(every nid appears individually, we have truncated 9000+ nids here) - RSV=%PreventMaint.2=; Other = -</td>
</tr>
</tbody>
</table>
States

**Production Uptime**

- **Up**
  - All evidence suggests that host is providing correct service.

- **Suspect**
  - Evidence of possible problems.

- **Unknown**
  - Insufficient information.

**Unscheduled Downtime**

- **HardDown**
  - Harshly terminated service.

- **SoftDown**
  - Gracefully terminated service.

**Scheduled Downtime**

- **SchedDown**
  - Expected hiatus in service.

---

<table>
<thead>
<tr>
<th>Operational State</th>
<th>SLURM</th>
<th>MOAB</th>
<th>CRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
<td>Up</td>
<td>Up</td>
<td>Up</td>
</tr>
<tr>
<td>Suspect</td>
<td>Draining</td>
<td>Down</td>
<td>Suspect</td>
</tr>
<tr>
<td>SoftDown</td>
<td>Drained, Drained*</td>
<td>Down</td>
<td>AdmIndown</td>
</tr>
<tr>
<td>HardDown</td>
<td>Draining*, Down</td>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td>SchedDown</td>
<td>(not tracked as a distinct state)</td>
<td></td>
<td>Down</td>
</tr>
</tbody>
</table>
Consider a 10k host supercomputer:

Track the state of each host separately, using the above state machine.
Transition Rules

“eventtype=cos_X-Y”
Defines a transition rule from X to Y.

Examples:

“cos_Up-Suspect=MOABUNSCHEDULEABLE | CRAYHEARTSTOPWARN | CRAYSUSPECT
If a host in Up emits any of these messages, count it as having transitioned to Suspect

“cos_HardDown-Up = MOABJOBSTART | MOABUNSCHEDULEABLE | CRAYBOOTED | CRAYAVAIL | CRAYHEALTHUP
If a host in HardDown emits any of these messages, count it as having transitioned to Up
## Transition Table

```
"cos_SchedDown-Unknown=MOABRSVSYSSEND"
```

That’s the only way out (ignore everything else, such as CRAYHEARTSTOPALERT)!

<table>
<thead>
<tr>
<th>New State</th>
<th>Event Type</th>
<th>Source</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
<td>MOABJOBSTART, MOABSCHEDULEABLE, CRAYBOOTED, CRAYAVAL, CRAYHEALTHUP</td>
<td>Moab, Syslog, Cray</td>
<td>A Job is starting. SchedDown deems the host to be available for scheduling. A host has completed its boot-up process. Cray management software deems the host as available for scheduling. The “node health checker” has set a compute host to the available state, e.g. after the host has entered a Suspect state and extensive checks were run on the system before risking starting additional jobs on it.</td>
</tr>
<tr>
<td>Suspect</td>
<td>MOABUNSC SCHEDULEABLE, CRAYHEARTSTOPWARN, CRAYSUSPECT</td>
<td>Moab, Cray, Syslog</td>
<td>Scheduler deems the host to be unavailable for new jobs. A host heartbeat has gone silent, indicating that the host might be dead. The “node health checker” has set a compute host to suspect because the host failed a health test.</td>
</tr>
<tr>
<td>SoftDown</td>
<td>CRAYADMIN DOWNS</td>
<td>Syslog</td>
<td>The “node health checker” (or system administrator) has marked a host as warranting additional attention before it should be trusted to run new jobs.</td>
</tr>
<tr>
<td>HardDown</td>
<td>CRAYHEARTSTOPALERT</td>
<td>Cray</td>
<td>A host heartbeat has gone silent for long enough that the host is considered to be dead, or there are directly observable signs that the host has died, such as well-known error codes in the heartbeat memory location. Cray management software will automatically kill dependent jobs, halt scheduling, or similar actions depending on the criticality of the declared dead host. See Table III for an example.</td>
</tr>
<tr>
<td></td>
<td>KERNEL_PANIC</td>
<td>Cray</td>
<td>A host kernel has panic’d (died). Most such events come through the Cray logs, there are are some hosts for which kernel panics can only be seen via syslogs.</td>
</tr>
<tr>
<td>SchedDown</td>
<td>MOABRSVSTARTSYS</td>
<td>Moab</td>
<td>A system reservation has started. Typically used to designate the start of a system maintenance period. See Table III for an example.</td>
</tr>
<tr>
<td>Unknown</td>
<td>MOABRSVENDSYS</td>
<td>Moab</td>
<td>A system reservation is complete.</td>
</tr>
</tbody>
</table>
**Disambiguated - CRAYHEARTSTOPALERT**

Red = “Failure”!
(A transition to Unscheduled Downtime)

Black = Benign!
(A normal reboot)

How else would you disambiguate the partial reboot and 1708 event?

![Graph.png](attachment://Graph.png)

- Nearly-weekly SchedDown’s
- 1708 concurrent host “failures”!
- A partial reboot during SchedDown
Summary Statistics

Call a CRAYHEARTSTOPALERT event indicating a transition to HardDown a “failure”:

Per Day

112 days (2/8-5/29)
69 perfect (no “failures”)
42 ok (1-5 “failures”)
1 bad (5/9, 1708 “failures” within several seconds)

Per Host

9231 hosts
7471 perfect (no “failures”)
1740 ok (1 “failures”)
19 questionable (2 “failures”)
1 bad (6 failures)

MTT“F”:

If we count 5/9 as a single “failure”:

\[
\frac{112 \text{ days} \times 9231 \text{ hosts}}{77 \text{ “failures”}} = 37 \text{ years/“failure”/host}
\]

If we count 5/9 as having 1708 “failures”:

\[
\frac{112 \text{ days} \times 9231 \text{ hosts}}{1784 \text{ “failures”}} = 1.6 \text{ years/“failure”/host}
\]

If we naively count all 70126 CRAYHEARTSTOPALERTS:

\[
\frac{112 \text{ days} \times 9231 \text{ hosts}}{1784 \text{ “failures”}} = 1.5 \text{ days/“failure”/host}
\]
Future Work

How about other event types? And other systems?

“Unknown” should be a start condition only - should instead track SchedDown:Ready and SchedDown:NotReady

How does this relate to job “failures”?

How does it help detection and prediction efforts?

What does it tell us about HPC fault/error/failure modes and distributions?

Interleaved red/black – probably missing some messages or eventtypes…

Here is the bad host btw

Oct’11-Sep’12 Sysadmin usage of app: monthly mean of 2078 non-interactive searches, median of 1755
For accurate understanding of our systems, event disambiguation is essential!

The Splunk “app” is called “HPC” and will appear at http://splunk-base.splunk.com/apps soon...

(extra slides follow)
Stack of Splunk Mechanisms

<table>
<thead>
<tr>
<th>Splunk Mechanism</th>
<th>Example Splunk-HPC app objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboards</td>
<td>DownHosts, NMTTI, SMTTI, JMTTI, ...</td>
</tr>
<tr>
<td>Saved Searches</td>
<td>hoststate, SystemHardDownAlert, ...</td>
</tr>
<tr>
<td>Custom Commands</td>
<td>stateMachine.py</td>
</tr>
<tr>
<td>Lookups</td>
<td>hostlist.py, hostnames.csv</td>
</tr>
<tr>
<td>Eventtypes</td>
<td>cos_HardDown-Up, OOM, MCE, ...</td>
</tr>
<tr>
<td>Indexes</td>
<td>hpc_SystemA, hpc_SystemB, ...</td>
</tr>
</tbody>
</table>