Debugging Openstack Problems
Using A State Graph Approach

Yong Xiang, Hu Li, Sen Wang, Charley Peter Chen and Wei Xu
Institute for Interdisciplinary Information Sciences (IIIS),
Tsinghua University
Modern systems are complicated
Modern systems are complicated (cont’d)
Trouble shooting for clients

My network is down!

User configuration? (attached NIC?)
Connected the Virtual Network to public?
Physical network down?
    OVS down?
    OVS agent down?
Network node down?
Floating IP not correctly configured?
Security group rules not set up correctly?
......
How many rules needs to know as a professional OpenStack operator?
The operational knowledge does not transfer!

Good news for IT consulting business.
Key idea: automatically discover knowledge in systems using most basic rules

• We can capture the knowledge: System Operation State Graph (SOSG)

• Turn ad-hoc system state queries into a uniform graph traversal.

• Anomaly detection to find hidden problems.
Data source used to construct the graph

- raw data sources, no semantic information

<table>
<thead>
<tr>
<th>Type</th>
<th>Data source</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>OpenStack databases updates (triggers)</td>
</tr>
<tr>
<td>Libvirt</td>
<td>libvirt status Python API</td>
</tr>
<tr>
<td>Ovs</td>
<td>OVS status ovsdb-client dump</td>
</tr>
<tr>
<td>Cephimage</td>
<td>Ceph image list rbd info</td>
</tr>
<tr>
<td>Cephfile</td>
<td>Ceph block file ls /ceph/dir/file</td>
</tr>
<tr>
<td>Cephlog</td>
<td>log files from Ceph</td>
</tr>
<tr>
<td>Log</td>
<td>logs from all OpenStack components</td>
</tr>
</tbody>
</table>

**States**

**Events**
How to construct the graph?

• Simple rule: discovery entity based on syntactic IDs
Can be very a large graph!

- 3-day operation data, about 40 GB
- Graph size: 43.3 million vertices, 56.6 million edges
How can we use the graph?

- System query as graph traversal
  - Ad-hoc queries => uniform method
  - No need to memorize tons of system-specific commands

- Anomaly detection
  - Automatically find the hidden problems in system with millions of states
System query as graph traversal

- If physical server A encounters a hard disk failure, which VMs are affected? (Ceph as the Openstack storage backend)

1. Which blocks are stored on the disk (Linux)
   `ls /var/lib/ceph/osd/…`

2. Which ceph image the block belongs to (Ceph)
   `rbd info -p compute(or volumes)`
   `rbd info -p compute(or volumes) <image>`
   `grep block-name-prefix filename`

3. Where the image is used (Openstack)
   `nova show <server>`
   `nova volume-show <volume>`
   `cinder show <volume>`
Anomaly detection: ideas

• Based on subgraph describing a single VM
  • find the subgraph that roots at the VM and also includes all its dependencies

• Distance-based anomaly detection
  • capture the structure information of the subgraphs in the distance metric
Anomaly case study: database record does not match physical states

- Subgraph: thousands of DB state vertices directly connect to the VM entity vertex
- Inspect: VM has been in deleted state for months, but the libvirt, Cephimage and OVS states still remain.
- Possible bug in retrying / recovery mechanism
Anomaly case study: failed VM migration

• Subgraph: the migrating VM is missing libvirt state, both from the source host and the destination host

• Inspect:
  • Database: nova.instance_faults shows “cannot remove config /etc/libvirt/...” on the source, and “error removing image” on the destination
  • Log: “instance not resizing, skipping migration” repeated 653 times

• Possible resource management bug
Future work

• Analyzing event and state history
  • Indicate why the system end up in an inconsistent state
  • Might predict the failure before actually happens
Future work

• Including other data sources
  • i.e. source code, bug reports and documentations
  • Help further confirm the proposed anomalies/bugs
  • Provide insights in how to fix the bugs discovered
Future work

• Supporting incremental graph construction
  • Capture the continuous evolving of state and event in a online way

• Applying SOSG to other systems
  • i.e. big data frameworks
  • General applicability
Conclusions

• System operation knowledge can be automatically discovered with simple rules

• Entities and links are important knowledge in systems, and can be captured with a state graph (SOSG)

• Many potential applications
  • System state query as graph traversal
  • Anomaly detection
  • …
Thank You

We are hiring: faculty members, postdocs in any CS field
contact: weixu@tsinghua.edu.cn