Installation

1. Prerequisites:
   - install ONE (versions above 5.4 operate with FaSS above v1.2, previous versions of ONE need FaSS v1.1 or before)
   - install InfluxDB and create fassdb
2. Install FaSS as root user:
   ```bash
   yum localinstall one-fass-service-v1.4-1.4.x86_64.rpm
   ```
3. Adjust the configuration file of the ONE scheduler, to allow it to point at the FaSS endpoint in `/etc/one/sched.conf`

Usage

1. Edit the initial shares for every user in `/tmp/one-fass/etc/shares.conf`
2. Start FaSS: `systemctl start fass`

High-level architecture

FaSS
- stateless (no duplicated info in DB)
- only handles queue
- asynchronous modules
- reorders queue
- kills pending VMs (synchronous)
- kills running VMs

Algorithms
- MultiFactor [2]

Default OpenNebula (ONE) scheduler is FIFO and based on static resources partitioning among the projects → not suitable for scientific DC

FairShare Scheduling (FaSS) service [1]

Task priorities assigned according to:
- v1.0 released with ElectricIndigo in April 2017 with ONE patch
- from v1.2 ONE 5.4 compatibility

Why scheduling fairly?

Large public cloud
- approximately infinite resources
- tenants are charged a posteriori
  ➔ applications scale in/out freely

Small scientific datacentre
- saturated regime
- tenants are charged a priori
  ➔ advanced resource allocation needed

Test with three users, initial shares configured in `/etc/fass/shares.conf`

<table>
<thead>
<tr>
<th>User</th>
<th>uid</th>
<th>gid</th>
<th>share</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huey</td>
<td>4</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>Dewey</td>
<td>5</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>Louie</td>
<td>6</td>
<td>1</td>
<td>25</td>
</tr>
</tbody>
</table>

Periodically instantiating VMs for all 3 users

Stress test

- Run for 1 week, ~22k VMs
- stable performance
- Adding periodical stress sources (0 – 300 VMs to random users)
- favours Huey
- after stress goes back to equilibrium

Performances

Equilibrated regime

Stress test

Keep your Cloud infrastructure clean with FaSS [3]
- Set VMs to be dynamic and to be terminated after a specific Time-to-Live (TTL)
- Instead of terminating, VMs can be powered-off, suspended or rebooted
- TTL settable per user.

Additional features

- Testing new algorithms
- Integrating into the production infrastructure

Stay tuned!

References: