Will it blend?
A comparison of oVirt, OpenStack® and kubernetes schedulers

Martin Sivák
Principal Software Engineer
Red Hat Czech

3th of Feb 2018
Agenda

Anatomy of a scheduler
- Goals
- Design considerations
- The three schedulers

Architecture similarities and differences
- Resource tracking
- Scheduling algorithm
- Balancing and preemption

Highlights and ideas to share
Goals of a scheduler

Find a place with enough resources to start the given VM[1] ...

... and make sure it keeps running
... and make sure it handles the load
... and keep the power consumption low
... and ...

[1] or container
Design considerations

- Size of cluster (~ hundreds of nodes)
- Deterministic algorithms
- Migrations and balancing
- Homogeneous cluster vs. heterogeneous cluster

- Pet vs. cattle
Scheduler as a function
The schedulers

oVirt

openstack

Kubernetes
## Number comparison

<table>
<thead>
<tr>
<th></th>
<th>oVirt</th>
<th>OpenStack</th>
<th>Kubernetes</th>
</tr>
</thead>
<tbody>
<tr>
<td>~ Max nodes</td>
<td>200</td>
<td>~300</td>
<td>5000</td>
</tr>
<tr>
<td>Language</td>
<td>Java</td>
<td>Python</td>
<td>Go</td>
</tr>
<tr>
<td>Load type</td>
<td>pet VMs</td>
<td>cattle VMs</td>
<td>containers</td>
</tr>
<tr>
<td>Resource tracking</td>
<td>pending + stats</td>
<td>placement service</td>
<td>pod spec in etcd</td>
</tr>
<tr>
<td>Active schedulers</td>
<td>1</td>
<td>1 or more</td>
<td>1 or more</td>
</tr>
</tbody>
</table>

**Notes:**
- `placement service` and `pod spec in etcd` refer to specific components or services used in OpenStack and Kubernetes, respectively.
- The `active schedulers` column indicates the number of active schedulers, which can be 1 or more for both OpenStack and Kubernetes.
Resource tracking
Resource tracking

**oVirt**

- pending resources are tracked, free resources come from reports

SIMPLIFIED!
kubernetes

- allocated resources are part of Pod spec,
  free = total - \( \sum \) spec
Resource tracking

OpenStack
- a placement service handles tracking and atomic resource reservation
The Algorithm
Algorithm - not rocket science

Filter
Remove all nodes that do not satisfy hard constraints

Map
Compute score, typically based on node load and free resources

Reduce
Select the best node
Filtering

Filter out incompatible nodes

Typical filters:
- CPU compatibility
- Free RAM
- Network presence
- Storage connectivity

Highlights:
- Affinity
- Load isolation and trust
- Labels
Scoring

Map a metric to a score
like CPU load 10% to 10.

Different metrics require different representation:
- CPU cores, running VM count - absolute number
- Free memory vs used memory - absolute or percents?
- CPU load vs “free” CPU - percents, something based on frequency? SMP?
- Label presence - boolean
Selecting the destination

Which node is the best? … it depends on the **goal**
- Maximizing performance, saving power or upgrade process?

Multiple metrics need **multipliers** for importance

So which node is the best then?
- How do you sum 10%, 3.5GiB and 16 together?
- **Normalization!**
Score normalization

<table>
<thead>
<tr>
<th>Project</th>
<th>Algorithm</th>
<th>To</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>oVirt</td>
<td>rank</td>
<td>-</td>
<td>compresses differences</td>
</tr>
<tr>
<td>OpenStack</td>
<td>scale / maximum over all hosts</td>
<td>0 - 1</td>
<td>depends on filter results</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>scale / single host</td>
<td>0 - 10</td>
<td>incorrect on heterogeneous clusters</td>
</tr>
</tbody>
</table>
Balancing and preemption
Balancing and Preemption

Methods
- offline migration (kill & re-start)
- preemption (kill & start other)
- live migration (move)

“Situations” emerging at runtime
- overload
- rule violations (eg. new affinity defined)

Selecting the best move
- select the object and select the move
- remember the deterministic assumption
- HARD!
Balancing - oVirt

Load balancing - equally balanced policy
Balancing - oVirt

Load balancing - power saving policy
Preemption - kubernetes

Can we kill low priority load when needed?

- Guaranteed load scheduling (DNS, network controller)
- Eviction policy (Help! I am overloaded)
- Disruption budget (Feel free to use one of mine)

Preemption in use elsewhere:

- AWS spot instances - money based priority
Highlights and good ideas
Interesting highlights

Scheduling:
- oVirt optimizer (probabilistic scheduling service)
- Chance scheduler (random selection)
- Arbitrary filtering rules in spec (booleans, operators)

Host devices:
- resource hierarchy and host device aliases

Resource tracking
- declarative and reactive - scheduler fills in data to Pod spec
Good ideas

- labels
- **normalization** methods
- **atomic** resource tracking and reservation

- multiple schedulers and split-brain protection
- balancing and preemption
Summary

All three schedulers are very similar in concept

Differences are small and based on the needs of the typical workload

There are ideas worth sharing!
THANK YOU!

Martin Sivák
msivak@redhat.com

with thanks to Red Hat’s OpenStack team