Solving (NP-Hard) Scheduling Problems with oVirt & OptaPlanner

Jason Brooks
Red Hat Open Source & Standards
SCALE13x, Feb 2015
What Is oVirt?

- Large scale, centralized management for server and desktop virtualization
- Based on leading performance, scalability and security infrastructure technologies
- Provide an open source alternative to vCenter/vSphere
- Focus on KVM for best integration/performance
- Focus on ease of use/deployment
Virt & Cloud Scheduling

- Running a new VM
- Selecting migration destination
- Load balancing
High Availability

- Build a highly available enterprise infrastructure
- Continually monitor host systems and virtual machines
- Automatically restart virtual machines in case of host failure
  - Restart virtual machine on another node in the cluster
- Use live migration to “fail-back” a VM to its original host when the server is restored
System Scheduler

- Dynamically balance workloads in the data center.
- Automatically live migrate virtual machines based on resources.
- Define custom policies for distribution of virtual machines.

Maintain consistent resource usage across the enterprise data center.
Define policies to optimize workload on a fewer number of servers during “off-peak” hours.
Scheduling in oVirt

<table>
<thead>
<tr>
<th>Host 1</th>
<th>Host 2</th>
<th>Host 3</th>
<th>Host 4</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>func 1</th>
<th>func 2</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Host 2</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Host 4</td>
<td>3</td>
<td>12</td>
</tr>
</tbody>
</table>

*Host 4 sum: 3*5+12*2 = 39
Filters

<table>
<thead>
<tr>
<th></th>
<th>func 1</th>
<th>func 2</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Host 2</td>
<td>10</td>
<td>2</td>
<td>54</td>
</tr>
<tr>
<td>Host 4</td>
<td>3</td>
<td>12</td>
<td><strong>39</strong></td>
</tr>
</tbody>
</table>

*Host 4 sum: 3*5+12*2 = 39
Weights

<table>
<thead>
<tr>
<th>Factor</th>
<th>func 1</th>
<th>func 2</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host 2</td>
<td>10</td>
<td>2</td>
<td>54</td>
</tr>
<tr>
<td>Host 4</td>
<td>3</td>
<td>12</td>
<td>39*</td>
</tr>
</tbody>
</table>

*Host 4 sum: 3*5+12*2 = 39
Balancers

- Triggers a scheduled task to determine which VM needs to be migrated to one of under-utilized hosts
- A single load balancing logic is allowed per cluster
External Scheduler

- External service written in python and run as a separate process from the engine
- External service provides:
  - Engine safety
  - Should allow additional languages
  - Future option of scheduling as a service
Optimizer Goals

- Better load balancing
- Configurable by existing cluster policy
- Separate machine to protect ovirt-engine
- Starting a VM that can't be placed directly
  - Space needs to be created first
Machine reassignment problem

- Defined by set of machines and set of processes
- Each machine has some resources (CPU, RAM, ...)
- Each process requires resources
- NP-complete (variant of bin packing)
  - Easy to verify a given solution to a problem in reasonable time.
  - There is no silver bullet to find the optimal solution of a problem in reasonable time (*).
Calculate the size of the search space

Given a Solution model, how many different combinations can it represent?

**Cloud balancing**

- **Model:** Computer → Process
- **Search space:** $c^p$

<table>
<thead>
<tr>
<th># computers</th>
<th># processes</th>
<th>search space</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100</td>
<td>8</td>
</tr>
<tr>
<td>200</td>
<td>600</td>
<td>$10^{1380}$</td>
</tr>
<tr>
<td>400</td>
<td>1200</td>
<td>$10^{5967}$</td>
</tr>
</tbody>
</table>

**Traveling salesman (TSP)**

- **Model:** linked list
- **Search space:** $n!$

<table>
<thead>
<tr>
<th># customers</th>
<th>search space</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>100</td>
<td>$10^{153}$</td>
</tr>
<tr>
<td>1000</td>
<td>$10^{1567}$</td>
</tr>
<tr>
<td>10000</td>
<td>$10^{15659}$</td>
</tr>
</tbody>
</table>

**Course scheduling**

- **Model:** Period → Room → Lecture
- **Search space:** $(p \times r)^t$

<table>
<thead>
<tr>
<th># periods</th>
<th># rooms</th>
<th># lectures</th>
<th>search space</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>64</td>
</tr>
<tr>
<td>36</td>
<td>6</td>
<td>100</td>
<td>$10^{233}$</td>
</tr>
<tr>
<td>36</td>
<td>18</td>
<td>400</td>
<td>$10^{1124}$</td>
</tr>
<tr>
<td>36</td>
<td>36</td>
<td>800</td>
<td>$10^{2490}$</td>
</tr>
</tbody>
</table>
Enter OptaPlanner

- optaplanner.org
- Optimization engine
- Many search algorithms
- Uses Drools Rule Language (DRL) for scoring
  - drools.org
Probabilistic approach

- Random search
  - Randomly generate a candidate solution
  - Evaluate and assign a score
  - Accept if better than the current
  - Rinse and repeat

- Smarter than random
  - Simulated annealing – closer and closer neighbors
  - Tabu search – do not repeat mistakes
OptaPlanner and oVirt
OptaPlanner and oVirt

- oVirt's Java-based policy units converted to DRL-based rules in order to honor admin-set filters and weights
  - not all policy units yet available through API
    - hosted engine score filters
    - CPU load-based balancing
- cluster info periodically acquired by the optimizer over oVirt's REST API, converted, and fed to the OptaPlanner's fact database
- performance is improved by caching all rule matches
- All previous facts and rules are then used together by the OptaPlanner solver engine to compute the result.
- The optimizer service keeps running and improving the solution.
- When something in the cluster changes, the facts update and the solver resumes using the current best solution as a base point.
Optimization steps

- Number of steps limited
- Slower to converge than simple “get me the optimum”
- Hard constraint check for each intermediate state
- Soft constraint check for the final situation only
# Web admin integration

![Web admin integration screen shot](image)

## SOLUTION STATUS

- **Status:** Solution received
- **()** Solution is being refreshed every 30 seconds.

## VMS THAT SHOULD BE STARTED

No VM starts are requested at this moment.

## MIGRATION / START STEPS

No migrations are needed at this moment. The state is stable.

## TARGET STATE

<table>
<thead>
<tr>
<th>host</th>
<th>vm</th>
<th>memory [all in GB]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>used</td>
</tr>
<tr>
<td><img src="image" alt="Host_two" /></td>
<td><img src="image" alt="aff_1" /></td>
<td>0.3</td>
</tr>
<tr>
<td><img src="image" alt="Host_three" /></td>
<td></td>
<td>0.3</td>
</tr>
</tbody>
</table>

---

Last Message: 2014-Sep-29, 13:25  User admin logged in
Error while executing action:

foo4:

- Cannot run VM. There is no host that satisfies current scheduling constraints. See below for details.
- The host ovirt-two did not satisfy internal filter Memory because it has insufficient free memory to run the VM.
- The host ovirt-one did not satisfy internal filter Memory because it has insufficient free memory to run the VM.

Physical Memory Guaranteed: 1500 MB
Number of CPU Cores: 1 (1 Socket(s), 1 Core(s) per Socket)
Guest CPU Count: N/A
Highly Available: No
Number of Monitors: 1
USB Policy: Disabled
Looking Ahead

- Tighter integration with BRMS
- Full automation of the optimization
  - Using the optimizer instead of the internal scheduler in oVirt engine
- Support for more Policy Units
  - Custom DRL rules
  - Units coming from external scheduler
- Long term cooperation potential
  - OpenStack Gantt
  - Kubernetes
  - Mesos
Questions?

http://www.ovirt.org
jbrooks@redhat.com
@jasonbrooks
jbrooks on OFTC & Freenode IRC