Applying Nova scheduler concepts for data center virtualization

Gilad Chaplik
Red Hat

FOSDEM, February 2014
What is oVirt

- oVirt is a KVM management application for data center virtualization management:
  - Manage hundreds of KVM hypervisor nodes, running thousands of VMs.
- Features:
  - Live migration of VMs and storage
  - Templates and snapshots of running VMs.
  - Advanced network configuration.
  - Support iSCSI, FCoE, NFS, and Gluster for shared storage.
Agenda

- The Need & Problem
- Nova Filter Scheduler concepts
  - Filter
  - Weights
- oVirt Scheduling
  - samples
- External Proxy
Re: [Users] How to define max number of running VMs on a host?

I have 4 graphic workstations with 3 graphic cards on each. I wanna passthrough graphic cards to the VMs one by one, since one workstation has only 3 cards, I must limit the number of running VM on a host to 3.
What we had..

- oVirt Scheduler
  - Executes the selected distribution algorithm on the cluster:
    - Even Distribution
    - Power Saving
  - Selects a host to run/migrate VM on.
  - Balance: Selects a VM to migrate and Host to migrate to.
  - Only 2 distribution algorithms, taking into consideration only CPU usage
  - No way to construct a user defined scheduling policy
The Filter Scheduler supports filtering and weighting to make informed decisions on where a new instance should be created.

- Filters
- Weights
Following Nova Scheduler

Source: http://docs.openstack.org/developer/nova/devref/filter_scheduler.html
Following Nova Scheduler

Source: http://docs.openstack.org/developer/nova/devref/filter_scheduler.html
class RamFilter(filters.BaseHostFilter):
    """Ram Filter with over subscription flag""

    def host_passes(self, host_state, filter_properties):
        """Only return hosts with sufficient available RAM."""
        instance_type = filter_properties.get('instance_type')
        requested_ram = instance_type['memory_mb']
        free_ram_mb = host_state.free_ram_mb
        total_usable_ram_mb = host_state.total_usable_ram_mb
        used_ram_mb = total_usable_ram_mb - free_ram_mb
        return total_usable_ram_mb * FLAGS.ram_allocation_ratio - used_ram_mb >= requested_ram
public cloud vs. data center virtualization

- Each Filter and Weight is applied on a single host
- Migration Domain
- Load balancing

- Implementation Details
  - Policy container
• Use internal/external Filters and Weights.
• Apply a method on all Hosts in Cluster.
• Containers - Cluster Policies.
• Load balancing for each cluster policy.
• Custom Properties.
### oVirt - The new Model

**Diagram:**
- Host 1 -> Host 2 -> Host 3 -> Host 4
- Host 1 -> Host 2 -> Host 4
- Host 2 -> Host 4

**Table:**

<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>54</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
• Filters Hosts of a given migration domain aka Cluster.
• Existing logic (pin-to-host, memory limitations, etc.) is translated into built-in Filters.
• External Filters written in python can be loaded into oVirt engine.
import datetime

class shut_down_hosts_filter:
    '''filters out hosts according to shutdown hour'''

    properties_validation = 'shut_down_hour=[0-9]*;wake_up_hour=[0-9]*;'

    def do_filter(self, hosts_ids, vm_id, args_map):
        wake_up_hour = int(args_map.get('wake_up_hour', -1))
        shut_down_hour = int(args_map.get('shut_down_hour', 25))
        current_hour = datetime.datetime.datetime.now().hour
        if(current_hour < wake_up_hour and current_hour > shut_down_hour):
            hosts_ids = []
        print(hosts_ids)
import datetime

class shut_down_hosts_filter():
    # Filters out hosts according to shutdown hour

    properties_validation = 'shut_down_hour=[0-9]*;wake_up_hour=[0-9]*;'

    def do_filter(self, hosts_ids, vm_id, args_map):
        wake_up_hour = int(args_map.get('wake_up_hour', -1))
        shut_down_hour = int(args_map.get('shut_down_hour', 25))
        current_hour = datetime.datetime.now().hour
        if(current_hour < wake_up_hour and current_hour > shut_down_hour):
            hosts_ids = []
        print(hosts_ids)
import datetime

class shut_down_hosts_filter:
    """filters out hosts according to shutdown hour"""

    properties_validation = 'shut_down_hour=[0-9]*;wake_up_hour=[0-9]*;'

    def do_filter(self, hosts_ids, vm_id, args_map):
        wake_up_hour = int(args_map.get('wake_up_hour', -1))
        shut_down_hour = int(args_map.get('shut_down_hour', 25))
        current_hour = datetime.datetime.now().hour
        if(current_hour < wake_up_hour and current_hour > shut_down_hour):
            hosts_ids = []
        print(hosts_ids)
```python
import datetime

class shut_down_hosts_filter:
    '''filters out hosts according to shutdown hour'''

    properties validation = {'shutdown hour': [0-23], 'wake up hour': [0-23]}

def do_filter(self, hosts_ids, vm_id, args_map):
    wake_up_hour = int(args_map.get('wake up hour', -1))
    shut_down_hour = int(args_map.get('shut down hour', 25))
    current_hour = datetime.datetime.now().hour
    if current_hour < wake_up_hour and current_hour > shut_down_hour:
        hosts_ids = []
    print(hosts_ids)
```
import datetime

class shut_down_hosts_filter:
    """filters out hosts according to shutdown hour"""
    properties_validation = 'shut_down_hour=[0-9]*;wake_up_hour=[0-9]*;'

    def do_filter(self, hosts_ids, vm_id, args_map):
        wake_up_hour = int(args_map.get('wake_up_hour', -1))
        shut_down_hour = int(args_map.get('shut_down_hour', 25))

        if(current_hour < wake_up_hour and current_hour > shut_down_hour):
            hosts_ids = []
        print(hosts_ids)
import datetime

class shut_down_hosts_filter:
    '''filters out hosts according to shutdown hour'''

    properties_validation = 'shut_down_hour=[0-9]*;wake_up_hour=[0-9]*;

    def do_filter(self, hosts_ids, vm_id, args_map):
        wake_up_hour = int(args_map.get('wake_up_hour', -1))
        shut_down_hour = int(args_map.get('shut_down_hour', 25))

        current_hour = datetime.datetime.now().hour
        if current_hour < wake_up_hour and current_hour > shut_down_hour:
            hosts_ids = []
        print(hosts_ids)
• Weights hosts that passed all filters.
• Predefined Weights:
  • Even Distribution
  • Power Saving
  • [3.4] VM Affinity, Even Guest Distribution, HA, etc.
• Factors.
• External Weights.
class even_vm_distribution():
    '''rank hosts by the number of running vms on them, with the least first'''

    properties_validation = ''

    def do_score(self, hosts_ids, vm_id, args_map):
        # open a connection to the rest api
        try:
            connection = API(url='http://host:port',
                             username='user@domain', password=''
            )
        except BaseException as ex:
            # letting the external proxy know there was an error
            print >> sys.stderr, ex
        return

        # get all the hosts with the given ids
        engine_hosts = \
            connectionhosts.list(
                query=''' or '''.join(['''id=%s'' % u for u in hosts_ids]))

        # iterate over them and score them based on the number of vms running
        host_scores = []
        for engine_host in engine_hosts:
            if(engine_host and
                engine_host.summary):
                host_scores.append((engine_host.id, engine_host.summary.active))
        print host_scores
class even_vm_distribution():
    # rank hosts by the number of running vms on them, with the least first
    properties_validation = ''

def do_score(self, hosts_ids, vm_id, args_map):
    # open a connection to the rest api
    try:
        connection = API(url='http://host:port',
                         username='user@domain', password='')
        print >> sys.stderr, ex
        return

    # get all the hosts with the given ids
    engine_hosts = \
        connection.hosts.list(
            query=" or " .join(["id=%s" % u for u in hosts_ids]))

    # iterate over them and score them based on the number of vms running
    host_scores = []
    for engine_host in engine_hosts:
        if(engine_host and
            engine_host.summary):
            host_scores.append((engine_host.id, engine_host.summary.active))
    print host_scores
class even_vm_distribution():
    '''rank hosts by the number of running vms on them, with the least first'''

    properties_validation = ''

    def do_score(self, hosts_ids, vm_id, args_map):
        #open a connection to the rest api
        try:
            connection = API(url='http://host:port',
                             username='user@domain', password='')
        except BaseException as ex:
            #letting the external proxy know there was an error
            print >> sys.stderr, ex
        return

        #iterate over them and score them based on the number of vms running
        host_scores = []
        for engine_host in engine_hosts:
            if(engine_host and
                engine_host.summary):
                host_scores.append((engine_host.id, engine_host.summary.active))

        print host_scores

        if(engine_host and
           engine_host.summary):
            host_scores.append((engine_host.id, engine_host.summary.active))

        print host_scores
Load Balancing

- Triggers a scheduled task to perform actions on a given cluster.
- A balancing logic can return a single VM, and oVirt Engine will migrate according to current scheduling logic.
- A single load balancing logic is allowed per cluster.
Load Balancing

- Predefined Load Balancing logic include:
  - CPU power saving
  - CPU even distribution
  - Even VM distribution
- External Balancing logic can be written in python and loaded into the engine.
```python
def do_balance(self, hosts_ids, args_map):
    wake_up_hour = int(args_map.get('wake_up_hour', -1))
    shut_down_hour = int(args_map.get('shut_down_hour', 25))
    current_hour = datetime.datetime.now().hour
    wake_up = current_hour > wake_up_hour and current_hour < shut_down_hour
    conn = self._get_connection()
    if conn is None:
        return
    for host_id in hosts_ids:
        host = conn.hosts.get(id=host_id)
        if (wake_up and host.status.state != 'up'):
            host.activate()
        elif (not wake_up and host.status.state != 'maintenance'):
            vms = conn.vms.list(query='host=' + host.name)
            for vm in vms:
                vm.stop()
            host.deactivate()
```
def do_balance(self, hosts_ids, args_map):
    wake_up_hour = int(args_map.get('wake_up_hour', -1))
    shut_down_hour = int(args_map.get('shut_down_hour', 25))
    current_hour = datetime.datetime.now().hour
    wake_up = current_hour > wake_up_hour and current_hour < shut_down_hour
    conn = self._get_connection()
    if conn is None:
        return

    vms = conn.vms.list(query='host=' + host.name)
    for vm in vms:
        vm.stop()
    host.deactivate()
def do_balance(self, hosts_ids, args_map):
    wake_up_hour = int(args_map.get('wake_up_hour', -1))
    shut_down_hour = int(args_map.get('shut_down_hour', 25))
    for host_id in hosts_ids:
        host = conn.hosts.get(id=host_id)
        if (wake_up and host.status.state != 'up'):
            host.activate()
        elif (not wake_up and host.status.state != 'maintenance'):
            vms = conn.vms.list(query='host=' + host.name)
            for vm in vms:
                vm.stop()
            host.deactivate()
        host.deactivate()
```python
selected_vm = None
# just pick the first we find
host_vms = connection.vms.list('host='+over_loaded_host.name)
if host_vms:
    selected_vm = host_vms[0].id
else:
    return

print (selected_vm, white_listed_hosts)
```
Cluster Policy

- Container for Filters, Weights and a single load balancing logic.
- Can be defined in oVirt Engine, and attached to a Cluster.
- Optimizations:
  - Speed
  - Overbooking
Cluster Policy – Define

![Cluster Policy Configuration Screen]

- Cluster Policies
  - Evenly_Distributed
  - None
  - Power_Saving
  - VM_Evenly_Distributed
Cluster Policy – Define

New Cluster Policy

Filter Modules
- Enabled Filters:
  - (EXT) shut_down_hosts_filter

Disabled Filters:
- VmAffinityGroups
- HA
- Network
- (EXT) shut_down_hosts_filter

Weights Modules
- Enabled Weights & Factors:
  - OptimalForEvenDistribution

Disabled Weights:
- None
- OptimalForEvenGuestDistribution
- VmAffinityGroups
- OptimalForPowerSaving

Load Balancer
- shut_down_hosts_balance

Properties
- wake_up_hour: 8
- shut_down_hour: 20
- No available keys
Cluster Policy – Define

**New Cluster Policy**

- **Name:** shut_down
- **Description:**

**Filter Modules** Drag or use context menu to make changes

- Enabled Filters:
  - [EXT]shut_down_hosts_filter

- Disabled Filters:
  - VmAffinityGroups
  - HA
  - Network
  - [EXT]shut_down_hosts_filter

**Weights Modules** Drag or use context menu to make changes

- Enabled Weights & Factors:
  - OptimalForEvenDistribution

- Disabled Weights:
  - None
  - OptimalForEvenGuestDistribution
  - VmAffinityGroups
  - OptimalForPowerSaving

**Load Balancer**

- shut_down_hosts_balance

**Properties**

- wake_up_hour: 6
- shut_down_hour: 20
- no_available_keys
Cluster Policy – Define

**New Cluster Policy**

**Filter Modules** Drag or use context menu to make changes

**Enabled Filters**

- (EXT) shut_down_hosts_filter

**Disabled Filters**

- VmAffinityGroups
- HA
- Network
- (EXT) shut_down_hosts_filter

**Weights Modules** Drag or use context menu to make changes

**Enabled Weights & Factors**

- OptimalForEvenDistribution

**Disabled Weights**

- None
- OptimalForEvenGuestDistribution
- VmAffinityGroups
- OptimalForPowerSaving

**Load Balancer**

- shut_down_hosts_balance

**Properties**

- wake_up_hour: 8
- shut_down_hour: 20
- No available keys
Cluster Policy – Define

**New Cluster Policy**

- **Name**: shut_down
- **Description**: 

**Filter Modules**

- **Enabled Filters**
  - (EXT)shut_down_hosts_filter
- **Disabled Filters**
  - VmAffinityGroups
  - HA
  - Network
  - (EXT)shut_down_hosts_filter

**Weights Modules**

- **Enabled Weights & Factors**
  - OptimalForEvenDistribution
- **Disabled Weights**
  - None
  - OptimalForEvenGuestDistribution
  - VmAffinityGroups
  - OptimalForPowerSaving

**Load Balancer**

- shut_down_hosts_balance

**Properties**

- wake_up_hour: 8
- shut_down_hour: 20
- No available keys
Cluster Policy – Define

New Cluster Policy

Name: shut_down

Filter Modules

Enabled Filters

(EXT) shut_down_hosts_filter

Disabled Filters

VmAffinityGroups
HA
Network
(EXT) shut_down_hosts_filter

Weights Modules

Enabled Weights & Factors

OptimalForEvenDistribution

Disabled Weights

None
OptimalForEvenGuestDistribution
VmAffinityGroups
OptimalForPowerSaving

Load Balancer

shut_down_hosts_balance (EXT)

Properties

wake_up_hour 8
shut_down_hour 20
No available keys

OK Reset Cancel
Cluster Policy – Attach to Cluster
External Scheduler

- Disabled by default (ExternalSchedulerEnabled = false)
- External service written in python and run as a separate process from the engine
- Why do we need it?
  - Engine safety
  - Should allow other languages
  - Going forward we may suggest SaaS (Scheduling as a Service)
External Scheduler

• Packaged as ovirt-scheduler-proxy RPM, which is optional (not installed by default).

• Initialization
  • Service Start
  • Analyze
  • Publishing Internal API (Starting XML-RPC Server)
  • Waiting for engine calls
    • Discover
    • ...

External Modules in oVirt

The image shows a configuration screen in oVirt with a list of external modules. The modules are listed under different categories such as system permissions, cluster policies, and others. The modules include:

- shut_down_hosts_filter
- shut_down_hosts_balance
- CPU
- CPU-Level
- HA
- Memory
- Network
- PinToHost
- VmAffinityGroups
- HA
- None
- OptimalForEvenDistribution

Each module has a 'Type' next to it, indicating whether it is a filter or a weight.
iterate over them and decide which to accept
accepted_host_ids = []
for engine_host in engine_hosts:
    if (engine_host and
        engine_host.summary.active < maximum_vm_count):
        accepted_host_ids.append(engine_host.id)
print accepted_host_ids
To sum it up..

- oVirt supports easy python plugins for VM scheduling
- Manage your scheduling policy separately for each migration domain (Cluster).
- New modules (features) are added in each version.
Questions?
THANK YOU!

http://www.ovirt.org
http://www.ovirt.org/Features/oVirtScheduler
http://www.ovirt.org/Features/oVirt_External_Scheduling_Proxy
http://www.ovirt.org/External_Scheduler_Samples

http://lists.ovirt.org/mailman/listinfo
vdsm-devel@lists.fedorahosted.org

#ovirt irc.oftc.net

gchaplik@redhat.com