HIGH-PERFORMANCE VMS

USING OPENSTACK NOVA

by Nikola Đipanov

$ WHOAMI

- Software engineer @ Red Hat
- Working on OpenStack Nova since 2012
- Nova core developer since 2013

THIS TALK

- OpenStack - the elastic cloud
- High-perf requirements in the cloud
- NUMA
- Large pages
- CPU pinning
- IO devices
- Challenge with exposing low level details in the cloud

OPENSTACK

Cloud infrastructure

Open-source (98.76% Python)

Multiple projects (compute, network, block storage, image storage, messaging, ...)

Self-service user API and dashboard (*aaS)

OPENSTACK NOVA

THE NOVA "ELASTIC CLOUD" APPROACH

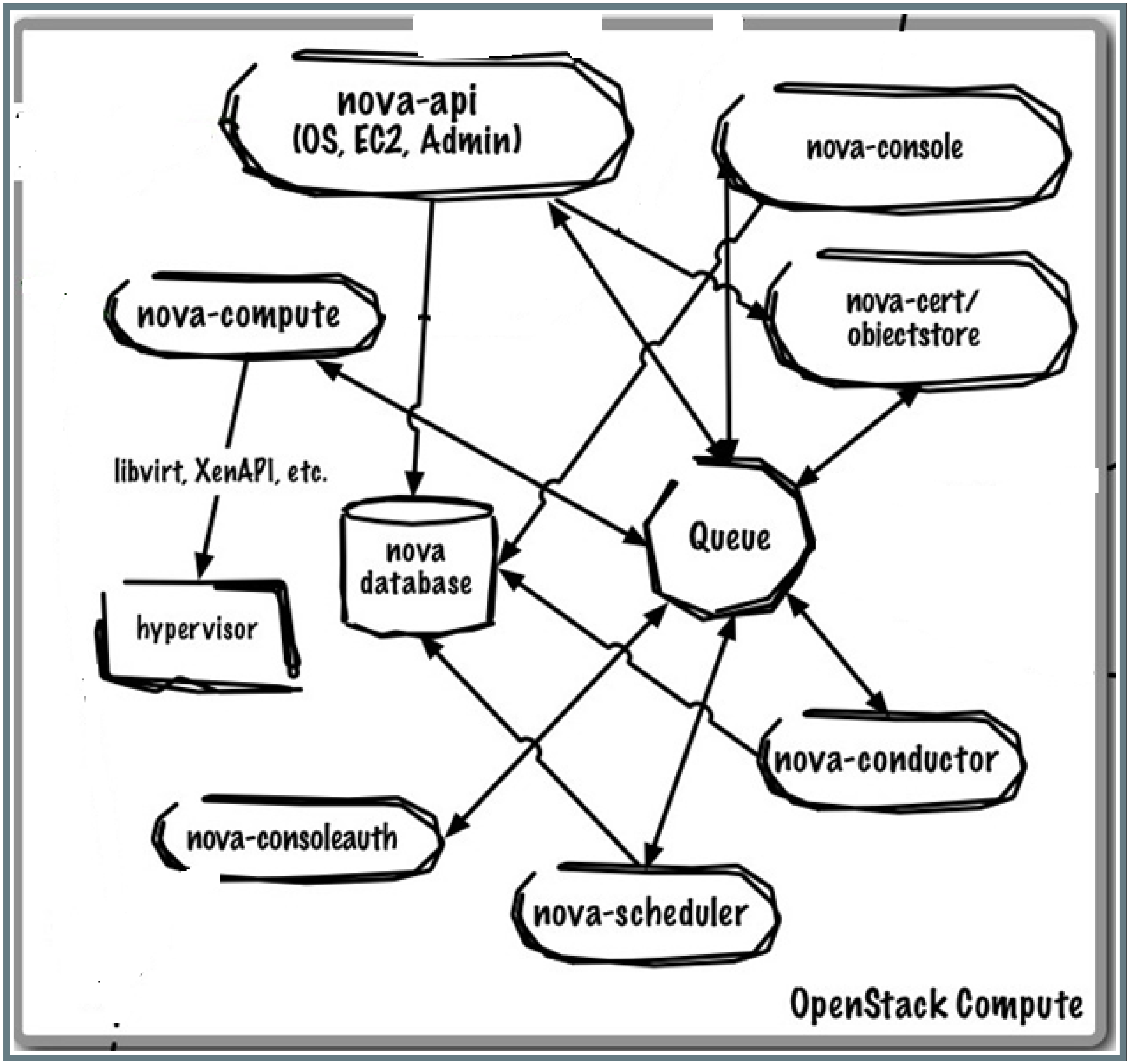
Allow for quick provisioning of new (comodity) hardware

Additional cloud resources (handled by other components) - VM images, block storage, networks...

Concept of *flavors* - combinations of VM resources (CPU, RAM, disk...)

Simple scheduling - focus on scale

Users have no visibility into hardware


nova-api
(OS, EC2, Admin)
nova-console
nova-compute
nova-cert/
obiectstore
libvirt, XenAPI, etc.
nova
database
Queue
hypervisor
nova-conductor
nova-consoleauth
nova-scheduler
OpenStack Compute

NOVA SCHEDULING (IN MORE DETAIL) 1/2

Flavor (admin controlled) has the basic information about resources assigned to an instance

Limited policy can be overriden through image metadata (mostly for OS/app related stuff)

Each compute host periodically exposes it's view of resources to the scheduler

For each instance request scheduler running each set of host resources through a set of filters

Considers only the ones that pass all filters (optionally in particular order)

NOVA SCHEDULING (IN MORE DETAIL) 2/2

Default filters consider overcommit of CPU/RAM (tunable)

Basic placement does not dictate how to use resources on the host granularity

(apart from PCI devs, kind of special cased)

HIGH-PERF REQUIREMENTS - MOTIVATION

Allow for performance-sensitive apps to run in the cloud

- Example use-case: Network Function Virtualization
  - Cloud instances with dedicated resources (a bit of an oxymoron)
  - The key is to allow for low (or at least predictable) latency

- Better HW utilization on modern machines
  - Have a way to take into account NUMA effects on moder hardware
  - Make this info available to the guest application/OS

HIGH-PERF REQUIREMENTS - THE CLOUD WAY

Relying on users having knowledge about the hardware they are running on - against the cloud paradigm

Need a way to allow users to request high-performance features without the need to understand HW specifics

NUMA AWARENESS

Modern HW increasingly providing NUMA

- Benefits of IaaS controller being NUMA aware:
  - Memory bandwith & access latency
  - Cache efficiency

- Some workloads can benefit from NUMA guarantees too (especially combined with IO device pass-through)
  - Allow users to define a virtual NUMA topology
  - Make sure it maps to actual host topology

NUMA - LIBVIRT SUPPORT (HOST CAPABILITIES)

```
<capabilities>
  <host>
    <topology>
      <cells num="2">
        <cell id="0">
          <memory unit="KiB">4047764</memory>
          <pages unit="KiB" size="4">999141</pages>
          <pages unit="KiB" size="2048">25</pages>
          <distances>
            <sibling id="0" value="10">
            <sibling id="1" value="20">
          </sibling></sibling></distances>
          <cpus num="4">
            <cpu id="0" socket_id="0" core_id="0" siblings="0">
            <cpu id="1" socket_id="0" core_id="1" siblings="1">
            <cpu id="2" socket_id="0" core_id="2" siblings="2">
            <cpu id="3" socket_id="0" core_id="3" siblings="3">
          </cpu></cpu></cpu></cpu></cpus>
```

REQUESTING NUMA FOR AN OPENSTACK VM

- Set on the flavor (admin only)
- Default - no NUMA awareness

- Simple case:
  - *hw:numa_nodes=2*

- Specifying more details:
  - *hw:numa_cpu.0=0,1*
  - *hw:numa_cpu.1=2,3,4,5*
  - *hw:numa_mem.0=500*
  - *hw:numa_mem.1=1500*

NUMA AWARENESS - IMPLEMENTATION DETAILS

- Compute host NUMA topology exposed to the scheduler
- Requested instance topology is persisted for the instance (NO mapping to host cells)
- Filter runs a placement algorithm for each host
- Once on compute host - re-calculate the placement and assign host<->instance node and persist it
- Libvirt driver implements the requested policy

NB: Users cannot influence final host node placement - it's decided by the fitting algo

NUMA LIBVIRT CONFIG - CPU PLACEMENT

```
<vcpu placement="static">6</vcpu>
<cputune>
  <vcpupin vcpu="0" cpuset="0-1">
  <vcpupin vcpu="1" cpuset="0-1">
  <vcpupin vcpu="2" cpuset="4-7">
  <vcpupin vcpu="3" cpuset="4-7">
  <vcpupin vcpu="4" cpuset="4-7">
  <vcpupin vcpu="5" cpuset="4-7">
  <emulatorpin cpuset="0-1,4-7">
</emulatorpin></vcpupin></vcpupin></vcpupin></vcpupin></vcpupin></vcp
```

NUMA LIBVIRT CONFIG - MEMORY AND TOPO

```
<memory>2048000</memory>
<numatune>
  <memory mode="strict" nodeset="0-1">
  <memnode cellid="0" mode="strict" nodeset="0">
  <memnode cellid="1" mode="strict" nodeset="1">
</memnode></memnode></memory></numatune>
<cpu>
  <numa>
    <cell id="0" cpus="0,1" memory="512000">
    <cell id="1" cpus="1,2,3,4" memory="1536000">
  </cell></cell></numa>
</cpu>
```

HUGE PAGES

Modern architectures support several page sizes

- Provide dedicated RAM to VM processes
- Maximize TLB efficiency

HUGE PAGES - SOME CAVEATS

- Need to be set up on the host separately (outside of scope of Nova)
  - This breaks the "commodity hardware, easily deployable" promise a bit
- VM RAM has to be a multiple of the page size
- No possibility for overcommit
  - Also interferes with the cloud promise of better utilization

REQUESTING HP FOR AN OPENSTACK VM

- Set on the flavor (admin only)
- Default - no huge pages
  - *hw:mem_page_size=large|small|any|2MB|1GB*

HUGE PAGES - IMPLEMENTATION DETAILS

- Each compute host exposes data about it's huge pages to the scheduler *per NUMA node*
- Filters run the same placement algorithm as fro NUMA, but now consider HP availability as well
- Once on compute host - re-calculate the placement and assign host<->instance node and persist it
- Libvirt driver implements the requested policy

HUGE PAGES LIBVIRT CONFIG

(Can be per node, but Nova does not allow that granularity)

```
<memorybacking>
  <hugepages>
    <page size="2" unit="MiB" nodeset="0-1">
    <page size="1" unit="GiB" nodeset="2">
  </page></page></hugepages>
</memorybacking>
```

CPU PINNING

- VM gets a dedicated CPUs for deterministic performance
- Improve performance of different workloads by avoiding/preferring hyperthreads.

CPU PINNING - SOME CAVEATS

- Requires a dedicated set of hosts (simple scheduling, no automatic VM reconfiguration)
  - This breaks the "commodity hardware, easily deployable" promise a bit too
- No possibility for overcommit (by design of course)
  - Trades off maximizing utilization for performance of specific workloads

REQUESTING HP FOR AN OPENSTACK VM

- Set on the flavor (admin only)
- Default - no CPU pinning

    - *hw:cpu_policy=shared|dedicated*
    - *hw:cpu_threads_policy=avoid|separate|isolate|prefer*
      proposed but not merged at this point

CPU PINNING - IMPLEMENTATION DETAILS

- Compute nodes expose available CPUs *per NUMA node*
- Filters run the same placement algorithm as for NUMA, but now consider CPU availability
- Flavors need to be set up to request for a specific set of hosts (an aggregate) in addition to the CPU pinning constraing
- Everything else same as for NUMA/HP

CPU PINNING LIBVIRT CONFIG

(memory is handled the same as for NUMA/Huge pages if requested)

```
<cputune>
  <vcpupin vcpu="0" cpuset="0">
  <vcpupin vcpu="1" cpuset="1">
  <vcpupin vcpu="2" cpuset="4">
  <vcpupin vcpu="3" cpuset="5">
  <vcpupin vcpu="4" cpuset="6">
  <vcpupin vcpu="5" cpuset="7">
  <emulatorpin cpuset="0-1,4-7">
</emulatorpin></vcpupin></vcpupin></vcpupin></vcpupin></vcpupin></vcp
```

PCI PASS-THROUGH DEVICE LOCALITY

- Pass-through of PCI devices (not developed as part of this effort)
- Make sure that PCI devices are local to the NUMA node the VM is pinned to

PCI DEVICE LOCALITY - IMPLEMENTATION DETAILS

- Compute nodes expose the NUMA node device is local too (libvirt has this info)
- Make sure that NUMA placement algo also considers requested PCI devices
- Current limitation - no matching of devices to guest nodes

HIGH PERF VMS IN OPENSTACK - THE GOOD PARTS

- Enable a major open source cloud solution to be used by a whole new class of users
- Expands the ecosystem, fosters innovation...

CHALLENGE WITH EXPOSING LOW LEVEL DETAILS IN THE CLOUD

- We cannot expose low level details to the user so the API needs to hide them while still being useful
- Complicates scheduling (SW) and hardware management (Ops)
- Nova specific challenges:
  - Not used by a big chunk of users - off by default
  - Internals (esp. scheduler) code not up to the complexity needed for it to work properly

QUESTIONS?

THANK YOU!

openstack.
CLOUD SOFTWARE

redhat.