OVN Testing and CI - an update

Dumitru Ceara, 2023
Agenda

- Overview of current OVN contributions
- In-tree “unit” and system tests
- End-to-end tests with ovn-kubernetes
- Control plane scale tests with ovn-heater
- Control plane scale tests with real CMS
- Data plane performance regression testing
- OVN CI: what, where, when?
- How do we improve all this?
Current OVN contributions

- OVN is still a very active project!
  
  \~30 features/user visible changes contributed in the last 12 months:

  $ last_year=$(git log --format=format:"%H" origin/main --since=11/26/2022 | tail -1)
  $ git diff $last_year..origin/main -- NEWS | grep '^+ -' -c

  30

  568 patches contributed in the last 12 months:

  $ git log --oneline origin/main --since=11/26/2022 --until=11/26/2023 | wc -l

  568

- For each patch developers, reviewers, maintainers must ensure:
  - it does what it’s supposed to (review/manual testing/etc)
  - the code quality is acceptable (review)
  - it doesn’t break existing functionality (CI)
  - it doesn’t introduce control plane scalability regressions (CI)
  - it doesn’t introduce data plane performance regressions (CI)

- Most patch series have more than one revision!
In-tree “unit” tests

$ make check TESTSUITEFLAGS="-l"
[...]  
806: ovn-ic.at:386  ovn-ic -- route sync -- ovn-northd -- parallelization=yes -- ovn_monitor_all=yes
807: ovn-ic.at:386  ovn-ic -- route sync -- ovn-northd -- parallelization=yes -- ovn_monitor_all=no
[...]  

In total **834 tests** split in:

- actual unit tests (hint: they use `ovstest` ... to run unit tests): ovn-features.at, ovn-ipam.at, ovn-lflow-cache.at, ovn-vif-plug.at, etc.
- system tests “in disguise” (hint: `ovn.at`):
  - often test complex scenarios (multiple simulated hypervisors, complicated logical topologies)
  - run in OVN sandboxes
  - use the OVS “dummy” datapath with the userspace conntrack implementation
  - workloads are simulated with dummy interfaces
  - some of these are run multiple times, varying arguments like:
    - `ovn-monitor-all` (yes/no)
    - `parallelization` (yes/no)*

* Most tests are run with parallelization=yes, the only exceptions are in `ovn-northd.at`.
In-tree system tests

In total **167 tests**:

- they bring up a real **single node** OVS and OVN environment
- 3 flavors of datapath:
  - kernel (netlink datapath)
  - netdev (userspace datapath)
  - dpdk (userspace datapath)
- workloads are actual veths running in network namespaces
- some of these are run multiple times, varying arguments like:
  - ovn-monitor-all (yes/no)

$ make check-kernel TESTSUITEFLAGS="-1"

[...]
147: system-ovn.at:11226 ovn mirroring -- ovn-northd -- parallelization=yes -- ovn_monitor_all=yes
148: system-ovn.at:11226 ovn mirroring -- ovn-northd -- parallelization=yes -- ovn_monitor_all=no
[...]
In-tree multi-node system tests

Just **one set of tests** for now:

- they bring up an OVN cluster running in containers (**ovn-org/ovn-fake-multinode**)
- configure a logical network topology and run traffic across different components of the network
- what is ovn-fake-multinode?
  - originally created by Numan Siddique to deploy “plain” OVN clusters with individual “fake” chassis running as containers
  - inspired from **kind** (kubernetes in docker)
  - each controller/compute node runs in its own container
  - each container looks like a “real” OVN chassis:
    - runs OVS
    - runs ovn-controller
    - central containers run NB/ovn-northd/SB
- Allows us to run upgrade-like tests (ovn-controllers running newer OVN versions, ovn-northd running older versions)

```
$ make check-multinode TESTSUITEFLAGS="-l"
[...]
1: multinode.at:3 ovn multinode basic test
[...]
```
End-to-end tests with ovn-kubernetes

**ovn-kubernetes is the community’s own kubernetes CNI plugin:** [https://github.com/ovn-org/ovn-kubernetes](https://github.com/ovn-org/ovn-kubernetes)

- Be nice people: try to ensure we don’t break ovn-kubernetes with OVN changes.
- Essentially, run a subset of ovn-kubernetes’ own tests on any OVN version (tree):
  - Build a custom ovn-kubernetes container image (with OVS/OVN compiled and patched from scratch).
  - Bring up a kubernetes cluster running in containers (kind) with ovn-kubernetes as CNI.
  - Run a (subset of) upstream kubernetes conformance tests:
    - `{"target": "shard-conformance", "ha": "HA", "gateway-mode": "local", "ipfamily": "ipv6", "disable-snat-multiple-gws": "snatGW"}`
    - `{"target": "shard-conformance", "ha": "HA", "gateway-mode": "local", "ipfamily": "dualstack", "disable-snat-multiple-gws": "snatGW"}`
    - `{"target": "shard-conformance", "ha": "HA", "gateway-mode": "shared", "ipfamily": "ipv4", "disable-snat-multiple-gws": "snatGW"}`
    - `{"target": "shard-conformance", "ha": "HA", "gateway-mode": "shared", "ipfamily": "ipv6", "disable-snat-multiple-gws": "snatGW"}`
  - Run a (subset of) upstream ovn-kubernetes control plane tests:
    - `{"target": "control-plane", "ha": "HA", "gateway-mode": "shared", "ipfamily": "ipv4", "disable-snat-multiple-gws": "noSnatGW"}`
    - `{"target": "control-plane", "ha": "HA", "gateway-mode": "shared", "ipfamily": "ipv4", "disable-snat-multiple-gws": "snatGW"}`
Control plane scale tests with ovn-heater

- Scale testing OVN control plane challenges:
  - testing a real cluster (OpenStack/Kubernetes) is complex: lots of CMS-specific knowledge required; bottlenecks identified in terms of CMS-specific components
  - not straightforward to find the right community/people to fix the bottleneck
  - harder to proactively implement a CI/CD pipeline from the control plane perspective: all tests can only be run whenever core OVN changes are consumed by the CMS (often that’s months later).

- https://github.com/ovn-org/ovn-heater “Mega script to install/configure/run a simulated OVN cluster deployed with ovn-fake-multinode.”
  - python & ansible based automation that relies on ovn-fake-multinode to set up and provision “fake” OVN nodes
  - contains its own collection of libraries to define workloads (uses python-ovs to configure the OVN databases)
  - integrated data collection (logs, perf traces, etc) and results interpretation (charts, latency stats, etc)
  - allows a faster test development pace for (new) OVN features
  - straightforward to use for CI/CD
Control plane scale tests with ovn-heater

- Rather specific “real world” workload simulations
  - together with CMS (cloud management system) contributors we defined scenarios that are interesting to simulate at scale:
    - ovn-kubernetes-like topologies: number of nodes, density of pods per node, number of services and service endpoints, etc
    - OpenStack-like topologies: number of computes/controllers, projects, external/internal networks, density of VMs, etc
  - came up with templated test scenarios implementing the above:

- **Note**: testing control plane scalability on public resources is not that easy; for reproducibility of results and full control of the deployment, ovn-heater tests run on bare-metal lab infrastructure downstream.

- it turned out to be a very prolific exercise from a performance perspective:
  - lots of improvements in OVN components (ovn-controller, ovn-northd)
  - lots of improvements on the ovsdb-server side (more details in Ilya’s OVSCon’22 presentation)

More info about ovn-heater in this [community meeting presentation](#) from earlier this year.
Control plane scale tests with real CMS (OpenShift)

- What better way to scale test how a CMS would use OVN than to actually deploy that CMS and run tests against it?
- Together with dedicated performance and scale teams we test OpenShift downstream at scale:
  - Bring up large clusters: 120, 250, 500, 1000 nodes
  - Run real workloads on these clusters: [https://github.com/cloud-bulldozer/kube-burner/tree/master/examples/workloads](https://github.com/cloud-bulldozer/kube-burner/tree/master/examples/workloads)
  - Collect relevant OVN information (logs, databases) and relevant metrics for the tested workloads.
- Pros:
  - Full fledged OpenShift => no potential gap between simulation and real CMS behavior.
- Cons:
  - Harder to integrate upstream.
  - Requires even more CMS specific knowledge when debugging/interpreting results.
  - More complex to set up than other control plane test tools (ovn-heater).
  - Usually triggered on demand.
- Note:
  - There are ways of deploying [OpenShift clusters on baremetal (in VMs)](https://github.com/cloud-bulldozer/kube-burner/tree/master/examples/workloads), with custom ovn-kubernetes images => it’s possible to create a CI pipeline to test real OpenShift at scale with vanilla upstream OVN code (WIP).
Data plane performance regression testing

- Periodic test runs with OVN-only cluster automation
- Various combinations of traffic patterns trying to replicate common kubernetes/OpenStack scenarios
- Really focused on the dataplane performance (no control plane tests):
  - hardware traffic generator -> a couple of OVN nodes (geneve) -> hardware traffic generator
  - test for TCP/UDP throughput and latency
**OVN CI: what, where, when?**

<table>
<thead>
<tr>
<th>Event</th>
<th>Who</th>
<th>Action</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PATCH posted to dev mailing list</strong> ([ovs-dev,v3] controller: Don't artificially ...)</td>
<td>ovsrobot/pw-ci</td>
<td>Apply patch to corresponding branch, run checkpatch, push series branch to <a href="#">ovsrobot ovn fork</a></td>
<td><a href="#">ovs-build</a> email reply</td>
</tr>
<tr>
<td><strong>commit pushed to ovsrobot/ovn fork</strong></td>
<td>ovsrobot/pw-ci</td>
<td>GitHub actions triggered</td>
<td>GitHub actions <a href="#">results</a></td>
</tr>
<tr>
<td><strong>ovsrobot fetches results of the GH actions</strong></td>
<td>ovsrobot/pw-ci</td>
<td>Parse GH actions result</td>
<td><a href="#">ovs-build</a> email reply (link to GH actions <a href="#">unit and system test results</a> and also <a href="#">ovn-kubernetes test results</a>)</td>
</tr>
<tr>
<td><strong>maintainer applies patch to OVN</strong></td>
<td>GitHub</td>
<td>GitHub actions triggered but <strong>also</strong> CirrusCI actions triggered (unit and system tests on ARM)</td>
<td>GitHub actions <a href="#">results</a>, CirrusCI <a href="#">results</a> Build badges (see bottom)</td>
</tr>
</tbody>
</table>

---

![Build and Test](passing) ![ovn-kubernetes](passing) ![System tests using ovn-fake-multinode](passing) ![build](passing) ![docs](passing)
## OVN CI: what, where, when?

<table>
<thead>
<tr>
<th>Event</th>
<th>Who</th>
<th>Action</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule (weekly)</td>
<td>GitHub</td>
<td>Run system tests with ovn-fake-multinode.</td>
<td>GitHub actions results</td>
</tr>
<tr>
<td>Schedule (weekly)</td>
<td>GitHub</td>
<td>Run unit and system tests (with OVS most recent stable branch).</td>
<td>GitHub actions results</td>
</tr>
<tr>
<td>Schedule (weekly)</td>
<td>GitHub</td>
<td>Build custom ovn-kubernetes container images with latest OVN main code.</td>
<td>GitHub actions results</td>
</tr>
<tr>
<td>Schedule (weekly)</td>
<td>Downstream automation</td>
<td>Trigger ovn-heater runs in Red Hat lab; matrix:</td>
<td>Potentially through emails, bug reports, upstream patches (no automated mechanism)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 20, 120, 250, 500 nodes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IPv4 &amp; IPv6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- density-light/heavy/cluster-density/network-policy tests</td>
<td></td>
</tr>
<tr>
<td>Schedule (weekly)</td>
<td>Downstream automation</td>
<td>Trigger dataplane tests in Red Hat lab to verify for</td>
<td>Potentially through emails, bug reports, upstream patches (no automated mechanism)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>potential performance (throughput/latency) regressions.</td>
<td></td>
</tr>
</tbody>
</table>
How do we improve all this? - stability

- Improve in-tree CI:
  - Detect unstable in-tree unit/system tests: avoid hiding failures via recheck
  - Fix unstable in-tree unit/system tests: **100+** commits to fix tests in the last 12 months
  - Pin versions of non-OVN components to avoid noise in testing: "ci: Pin Python, Fedora and Ubuntu runner versions." and "Allow to use different container images per branch."
  - Improve complex and slow tests (e.g., use fmt-pkt more often now that it’s becoming fast)
- Use stable ovn-kubernetes releases(*) in ovn-org/ovn CI (avoid noise from bleeding-edge code in ovn-kubernetes)

(*) ovn-kubernetes has no stable releases at the moment
How do we improve all this? - coverage

- Expand control plane scale testing (ovn-heater):
  - Add support for running OVN-IC clusters with ovn-heater (*initial work is merged*, last part is *under review*)
  - Add OVN-IC tests to downstream test matrix
  - Add more OpenStack scenarios
  - Automate result reporting downstream -> upstream

- Expand the data plane testing infrastructure:
  - Add more traffic patterns
  - Automate result reporting downstream -> upstream

- Complete downstream CI pipeline to deploy OpenShift/kubernetes at scale on baremetal with vanilla upstream OVN:
  - Run control and data plane CMS specific tests
  - Automate result reporting downstream -> upstream
Questions, suggestions, feedback?
Thank you!