Evaluate NFVI Performance by Leveraging OPNFV Testing Projects

Gao Liang, Huawei
Trevor Cooper, Intel Corp.
Agenda

1. Challenges
2. Rules & Choices
3. Examples & Demo
4. Future Directions
NFV Performance Testing Challenges

- Complex SUT configurations are required for optimal performance
- Numerous choices are required for test tools as well as test environments
- Virtual applications and network NFVI resources may interact
- Multi-vendor solutions add to testing complexity
- Deployment/orchestration automation impacts performance
- Testing methods must be integrated with CI and DevOps

Traditional Performance Testing Methods are Useful but Not Sufficient
“Rules” for Efficient NFV Performance Testing

1. Know yourself ... developer, solution architects, user perspectives differ

2. Know your use-case ... service and network SLAs define workloads

3. Know your words ... metric definitions and test methods impact results

4. Know your friends ... OPNFV and other communities can help you
OPNFV Test Ecosystem

OPNFV Testing Community

Compliance

Functional Testing

Database Test Results

dovetail

OPNFV Compliance Verification

NPV, VIM, MANO-APN & Functions

functest

Feature Projects

Performance Testing

TestDPI

Database Analytics & Dashboards

Database Test-cases & Results

Scenario Status Webpage

Test API

Feature Projects

Performance Testing

Pharos LF & Community Lab Infrastructure

Dev Resources

CI Integration Resources

CI Test Resources

Test Tiers

In Service

Stress

VNF

Performance

Components

Features

Smoke Test

Health Check
### OPNFV Performance Testing Menu

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NFVI + VIM + MANO</strong></td>
<td><strong>Commercial hardware</strong></td>
<td><strong>NFVI + VIM + MANO</strong></td>
</tr>
<tr>
<td>- SampleVNF (vACL, vFW, vCGNAT, …)</td>
<td>- Ixia</td>
<td>- Dovetail</td>
</tr>
<tr>
<td>- SFC, …</td>
<td>- Spirent</td>
<td>- NFVbench</td>
</tr>
<tr>
<td><strong>NFVI + VIM</strong></td>
<td><strong>Commercial virtual</strong></td>
<td><strong>NFVI + VIM</strong></td>
</tr>
<tr>
<td>- Compute</td>
<td>- Ixia</td>
<td>- Yardstick</td>
</tr>
<tr>
<td>- Storage</td>
<td>- Spirent</td>
<td>- Storperf</td>
</tr>
<tr>
<td>- Network</td>
<td></td>
<td>- Bottleneck</td>
</tr>
<tr>
<td>- vloop-vnf (dpdk-testpmd, Linux bridge, L2fwd module)</td>
<td></td>
<td>- Qtip</td>
</tr>
<tr>
<td>- Spirent stress-VM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Virtual Traffic classifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NFVI</strong></td>
<td><strong>Open Source software</strong></td>
<td><strong>NFVI</strong></td>
</tr>
<tr>
<td>- Virtual switching (OVS, OVS-dpdk, VPP)</td>
<td>- Pktgen</td>
<td>- VSPERF</td>
</tr>
<tr>
<td>- HW offload (TSO, crypto, SmartNIC)</td>
<td>- Moongen</td>
<td>- Storperf</td>
</tr>
<tr>
<td>- Physical / virtual interfaces</td>
<td>- TREX</td>
<td></td>
</tr>
<tr>
<td>- NIC (10GE, 40GE, …)</td>
<td>- PROX</td>
<td></td>
</tr>
<tr>
<td>- Vhost-user</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Pass-through, SR-IOV</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Drive standardization**

**Promote a defined platform and reuse**

**Establish best practice**
Euphrates Performance Testing

- NFVI components & subsystems
- Project Master (N) Merge & Verify
- Project Stable (N-1) Daily

Test Artifacts
- Test Scripts
- Test Cases
- Test Data
- Test Dashboards

Configurations
- Performance Ranges

OPNFV Dev
CI/CD/CT

OPNFV Production
CI/CD/CT

Pre-production Evaluation
- Operator Proprietary Tests
- Commercial VNF Testing
- User NFVI Testing

Opnfv Production
CI/CD/CT

Release testing

Dashboards

Reference
Performance Ranges

Reference
Configurations

Vendor testing
- Operator testing

Developer testing
VSPERF DUT is an important part of the E2E Data Path

- Virtual Switching technology and offloads
- Physical and virtual ports
- Virtualized Workload

VSPERF Test Automation

- Source/build SW components
- Set up vSwitch
- Set up workload
- Set up traffic generator
- Execute test cases
- Collect test results
- Log and store data
- Generate test statistics & result dashboards / reports

Network SLA
- Capacity / BW
- Loss
- Delay
- Delay variation

Network Performance Metrics & Statistics
VSPERF Dataplane Performance Testing

Switching Operations

- Speed
- Accuracy
- Reliability
- Scalability

NFVI Forwarding Capability

- Frame loss
- Maximum forwarding rate
- Burst behavior
- Packet Delay Distribution
- Scalability (flows, active ports, etc.)
Yardstick Overview

- Methodology for verifying infrastructure from the perspective of a VNF aligned with **ETSI TST 001**.

- Where do we fit

![Diagram](image)
### Yardstick Test case Overview

<table>
<thead>
<tr>
<th>Network</th>
<th>Compute</th>
<th>Storage</th>
<th>NFVI Generic</th>
<th>NFV extended</th>
<th>Network Service Benchmark</th>
<th>NFV extended</th>
<th>Integrating test case</th>
<th>Feature Test Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Network Latency between VM&lt;br&gt; * Network throughput and packet loss&lt;br&gt; * Packet delay variation&lt;br&gt; * Network throughput and packet loss&lt;br&gt; * Network latency in dpdk&lt;br&gt; * Network latency between nodes&lt;br&gt; * Cache hit/miss ratio and usage, nw thrp and latency&lt;br&gt; * Network capacity and scale&lt;br&gt; * IP datagram error rate, ICMP message error rate, TCP segment error rate and UDP datagram error rate</td>
<td>Cache utilization memory read latency memory rw bandwidth Processing speed, CPU Load Meminfo, CPUinfo</td>
<td>Storage IOPS thrp, latency disk size, block size and disk utilization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HA</td>
<td>Openstack Service HA&lt;br&gt; Neutron Server&lt;br&gt; Keystone&lt;br&gt; Glance Api&lt;br&gt; Cinder Api&lt;br&gt; Swift Proxy&lt;br&gt; Network HA&lt;br&gt; CPU Overload HA&lt;br&gt; Disk I/O Block HA&lt;br&gt; Load Balance Service HA&lt;br&gt; Openstack Virtual IP HA</td>
<td>Parser</td>
<td>translate from yang to tosca</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPv6</td>
<td>IPV6 network latency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KVM4NFV</td>
<td>KVM Latency measurements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vasperf</td>
<td>rfc2544_back2back_dummy&lt;br&gt; rfc2544_continuous_dummy&lt;br&gt; rfc2544_throughput_dummy</td>
<td>Bottlenecks</td>
<td>Posca stress traffic (throughput)&lt;br&gt; Posca stress ping (life cycle)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storperf</td>
<td>Storage performance via Storperf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Yardstick Architecture Overview

Extensible
- Restful API & CLI support, can be easily integrated
- Plug-in support, can easily integrate other tools
- Python 3 support

Customizable
- Core as a Orchestrator
- Scenarios as a library (Test operation)
- Contexts as a library (Test Pre-condition)
- Jinja2 support (Test case)

Ease to Use
- Docker image
- Offline support
- UI & DB and Report

[Diagram of Yardstick Architecture]
Yardstick Demo

• Using Yardstick test cases to evaluate NFVI performance
  • Installation
  • Web UI
  • SUT management
  • Test case management
  • Project->Task->report
  • Live Migration Test case Run

• Upload your Yardstick test case to evaluate NFVI performance
  • Upload test case

• Run storperf test case via Yardstick.
Future

• Advanced Stress Testing
• Simulator capabilities
• Post processing/analytics/staging
  • Data + modeling
• Performance Testing as a Service
  • Micro Service Approach
    • Run any test case from any projects you want with **Unified User Interface**
    • Common Traffic Generator, VNF on boarding, System level configuration.
    • All projects with Common DB, Test API, Dashboard