Red Hat OpenShift on IBM Z - Performance Experiences, Hints and Tips

Marc Beyerle (marc.beyerle@de.ibm.com)
Senior Java Performance Engineer, IBM Mainframe Specialist

Document version: 1.1
Document date: 2020-10-20

Notices and disclaimers

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers, *continued*

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and IBM Z, IBM z15, and z/VM are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml

Agenda

- Introduction
- Performance measurement and tuning approach
- Observations and recommendations
  - CPU-intensive workloads
  - Network-intensive workloads
- General tips for cloud-native applications
- Summary

Introduction

- Red Hat® OpenShift® is a ***hybrid cloud***, enterprise ***Kubernetes®*** platform
  - OpenShift website: *"…includes security, performance, and defect fixes, validated and tested integrations for third-party plugins, and enterprise lifecycle support"*
- Since it's a ***new technology*** for many IBM Z® customers, it brings its own set of requirements, pitfalls, etc.
  - Normal for new technologies
- ***Question***: Why is network performance ***important*** for applications running on OpenShift?
- ***Answer***: The general expectation is that most applications on OpenShift will be ***cloud-native*** type of applications, which are typically ***very network-sensitive***.
  - Highly componentized, built on ***micro-services***, ***distributed*** across the cluster
  - What used to be a ***function call*** for traditional applications is often a ***REST*** call now

Agenda

- Introduction
- Performance measurement and tuning approach
- Observations and recommendations
  - CPU-intensive workloads
  - Network-intensive workloads
- General tips for cloud-native applications
- Summary

High-level environment setup

- IBM z15™
- z/VM® LPAR
  - 3 master nodes
  - 5 worker nodes
- Bastion server LPAR
  - Software load balancer
- uperf target LPAR
  - Simulating an external system

- IBM DS8950F
- ECKD disks
- Model 9s for z/VM
- *Extended Address Volumes* (EAVs) for z/VM guests
- 5 x HyperPAV aliases for each z/VM guest

Approach

- First step was to install a ***basic OpenShift cluster*** with 3 master nodes and 5 worker nodes
  – First OpenShift version was 4.2.x, which got upgraded many times
- Got familiar with both the OpenShift web console and the ***Command Line Interface*** (CLI)
  – For "serious" work like scripting, etc., the CLI is much better suited
- Ran a selection of ***microbenchmarks*** that we typically use for a Linux® distribution evaluation
  – CPU-intensive, network-intensive, disk I/O-intensive, crypto-intensive, etc.
- Pre-requisite for running microbenchmarks: ***containerize*** workloads
  – In OpenShift, every application runs in one or more container(s)
- Used our Linux distribution evaluation results as a ***baseline*** for comparison
  – When doing performance work, a solid baseline is extremely important

Agenda

- Introduction
- Performance measurement and tuning approach
- Observations and recommendations
  - CPU-intensive workloads
  - Network-intensive workloads
- General tips for cloud-native applications
- Summary

Disclaimer

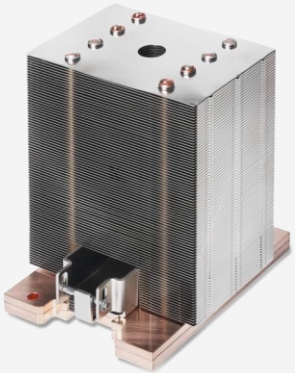
- The following is ***important*** – please read it carefully

- The performance test results in the following charts were obtained in a ***controlled lab environment***. The measured differences in performance – latency, throughput, etc. – might not be observed in real-life scenarios and customer production environments.

- All test runs were performed with IBM z15, z/VM 7.1, OpenShift 4.5.8, Red Hat® Enterprise Linux® CoreOS release 4.5, and uperf 1.0.3-beta. ***Other*** products and / or other product ***versions*** might produce ***different*** performance results.

CPU-intensive workloads

- Ran a selection of both ***CPU*** and ***memory-intensive*** workloads
  – Mostly Java®-based workloads
- Outcome: virtually ***no overhead*** compared to Linux distro baseline
  – Great result – expectation was to see at least a small degradation
- No need for further investigation



Agenda

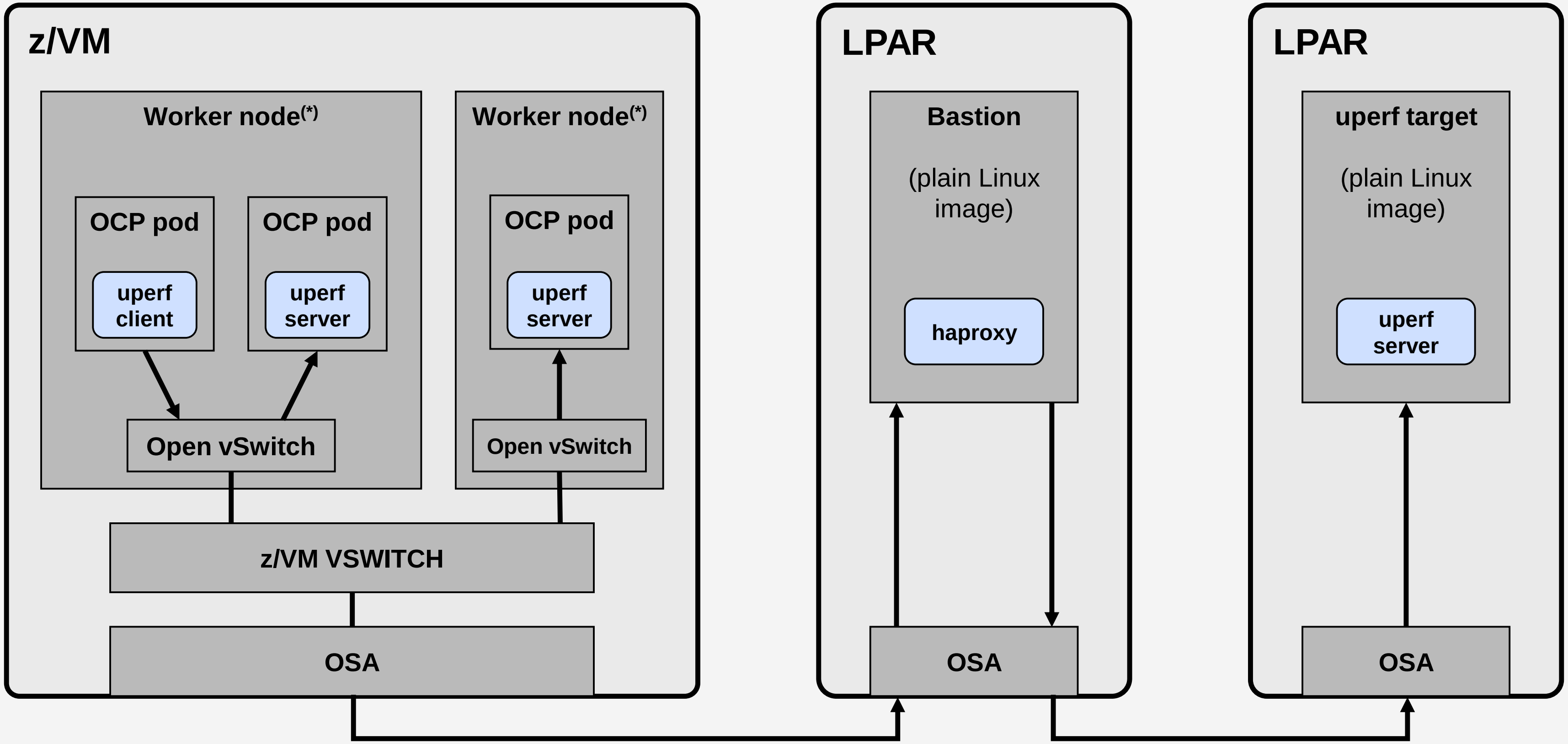
- Introduction
- Performance measurement and tuning approach
- Observations and recommendations
  - CPU-intensive workloads
  - Network-intensive workloads
- General tips for cloud-native applications
- Summary

Network-intensive workloads

- First impression: network performance results were ***below our expectations***
  - Caveat: in the beginning, we did not understand OpenShift network architecture well enough

- Had to invest ***quite some time*** in order to get familiar with OpenShift network concepts
  - Recommendation: set some time aside to read through the OpenShift online documentation

OpenShift network performance: the big picture



(*)For the sake of simplicity, this overview diagram only contains the most relevant components. In reality, there are many more.

Some OpenShift network concepts

- OpenShift makes heavy use of the ***Software Defined Network*** (SDN) approach
  – One physical network device per pod would be way too much – virtualized network is required

- Every ***pod*** has its own ***virtual*** network interface
  – Inside the pod, this is called `eth0`

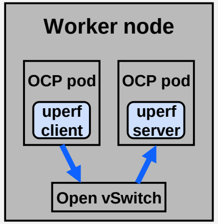
- This virtual network interface is connected to an ***Open vSwitch*** – not to be confused with z/VM VSWITCH
  – One Open vSwitch instance running per node

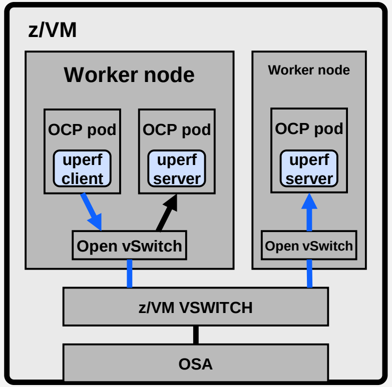
- Every Open vSwitch instance is configured based on ***OpenFlow™*** rules
  – Rules determine what packets are flowing where in the virtual network

- Network traffic that goes ***in and out of the cluster*** must pass through a ***load balancer*** of some form, which is the ***Bastion*** server in our test cluster
  – Load balancer can be either software-based or hardware-based

Network performance test scenarios

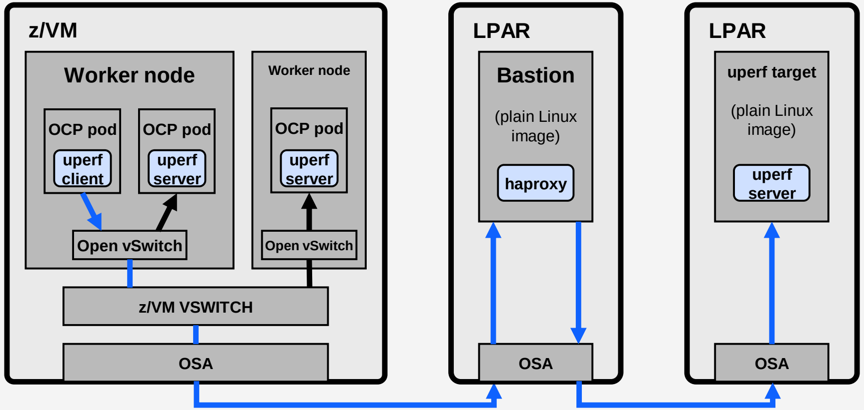
- Scenario #1: pod-to-pod communication ***inside the same*** worker node
  – Network traffic at the Open vSwitch layer only





- Scenario #2: pod-to-pod communication ***across*** worker nodes
  – In addition to scenario #1, z/VM VSWITCH is being used

- Scenario #3: pod-to-external communication
  – Traffic ***leaves*** the OpenShift cluster
  – All network layers are involved



Network performance evaluation basics

- Focus on ***latency*** and / or ***throughput***
  - Different requirements – high throughput doesn't necessarily mean low latency

- ***Small*** number of parallel connections versus ***large*** number of parallel connections
  - Single connection scenarios like backup, for example, versus online banking

- ***Transactional*** workloads ("ping-pong" type of communication) versus ***streaming*** workloads
  - Small network packets versus large or very large network packets

- A ***thorough*** analysis should consider either all or at least most of the above aspects

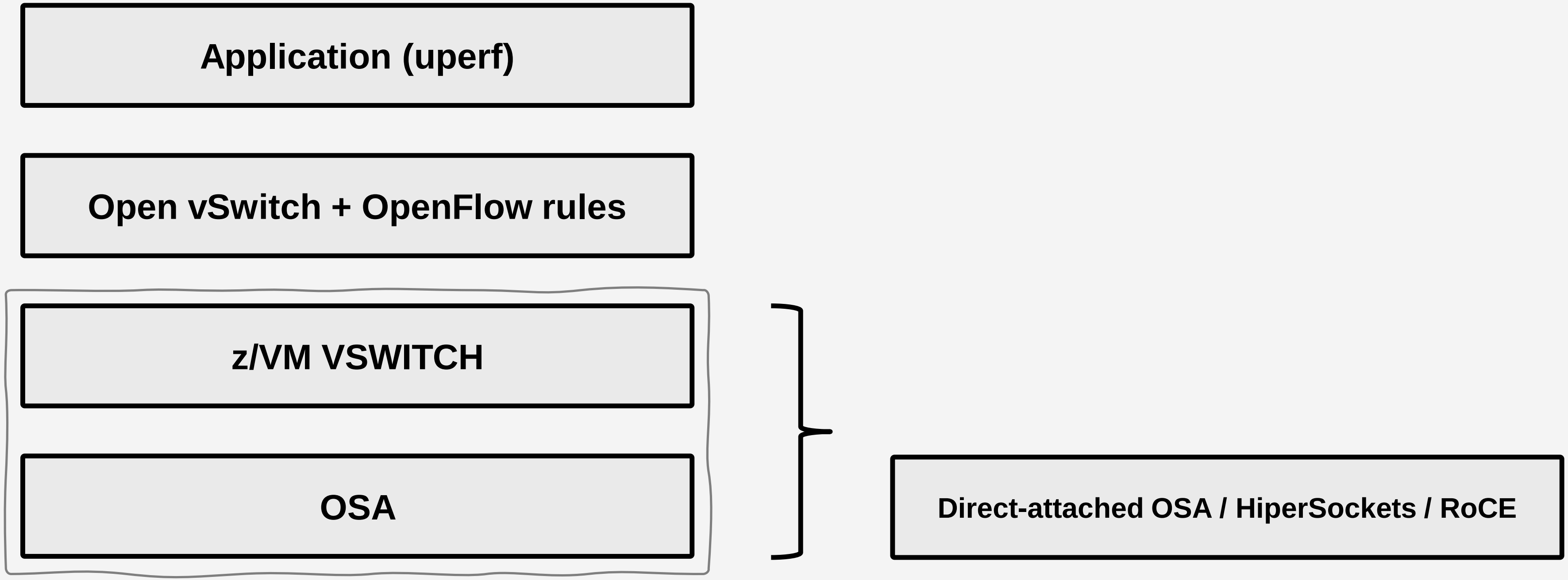
Network performance tuning

- Our goal: ***improve*** OpenShift network performance and give both easy-to-understand and easy-to-implement ***tuning hints and tips*** to existing and prospective customers

- Tuning category #1: change network configuration at the ***hardware level***
  - Exchange z/VM VSWITCH with direct-attached OSA card, HiperSockets, or RoCE card

- Tuning category #2: find ***configuration parameters*** that improve network performance
  - Much more complex, since there are literally hundreds of different configuration settings spread across the entire ***network*** and ***virtualization*** stack
  - Requires ***deep technical knowledge*** of different components of overall architecture (both virtualization and network)

Tuning category #1: hardware alternatives



Tuning category #2: network tuning options

- ***Finding*** and ***testing*** the below mentioned parameters and settings required an ***enormous*** amount of effort
  – Had to verify whether all scenarios either benefit or are at least not impacted

- Tuning recommendation #1: ***Receive Packet Steering*** (RPS)
  – Allows multiple CPUs to participate in network packet processing

- Tuning recommendation #2: adding one or more ***static route(s)*** for external traffic
  – Depends on your network architecture

- Tuning recommendation #3: move ***network-intensive*** OpenShift components away from your regular worker nodes out to ***infrastructure nodes***
  – Remedies performance imbalance / performance fluctuations across worker nodes

Tuning recommendation #1: Receive Packet Steering (RPS)

- RPS is controlled via a Linux ***sysfs*** entry called `rps_cpus`
  - Contents is a CPU mask
- Turned ***off***:

```
[root@worker-0 ~]# cat /sys/devices/qeth/0.0.6000/net/enc6000/queues/rx-0/rps_cpus
00000000,00000000
```

- Turned ***on***:

```
[root@worker-0 ~]# cat /sys/devices/qeth/0.0.6000/net/enc6000/queues/rx-0/rps_cpus
00000000,000000ff
```

- Can be changed via a ***udev*** rule, for example

Tuning recommendation #1: Receive Packet Steering (RPS), *continued*

- Create a plain text file called `enable-rps.txt` with the following contents:

```
# turn on Receive Packet Steering (RPS) for all network interfaces
ACTION=="add", SUBSYSTEM=="net", KERNEL=="*", ATTR{queues/*/rps_cpus}="ff"
```

- ***Question***: How can I apply this to all my worker nodes?

- ***Answer***: Make use of a ***Machine Config Object*** (MCO).

- Rule text has to be ***encoded*** like so:

```
cat enable-rps.txt |
python3 -c "import sys, urllib.parse; print(urllib.parse.quote(''.join(sys.stdin.readlines())))"
```

Tuning recommendation #1: Receive Packet Steering (RPS), *continued*

- Create a YAML file called `enable-rps.yaml` with the following contents:


```yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 50-enable-rps
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=US-
ASCII,%23%20turn%20on%20Receive%20Packet%20Steering%20%28RPS%29%20for%20all%20network%20interfaces%0A
ACTION%3D%3D%22add%22%2C%20SUBSYSTEM%3D%3D%22net%22%2C%20KERNEL%3D%3D%22%2A%22%2C%20ATTR%7Bqueues/%2A
/rps_cpus%7D%3D%22ff%22%0A
        filesystem: root
        mode: 0644
        path: /etc/udev/rules.d/71-net-tunings.rules
```

Tuning recommendation #1: Receive Packet Steering (RPS), *continued*

- Create a new MCO by running `oc create -f enable-rps.yaml`
- Double-check whether the MCO has been created: `oc get machineconfigs`
- Sample output on our test cluster:

```
NAME                                                        GENERATEDBYCONTROLLER                      IGNITIONVERSION   CREATED
00-master                                                   fac5a23280e6467ec95bb0237fae3ce387d04f0b   2.2.0             54d
00-worker                                                   fac5a23280e6467ec95bb0237fae3ce387d04f0b   2.2.0             54d
01-master-container-runtime                                 fac5a23280e6467ec95bb0237fae3ce387d04f0b   2.2.0             54d
01-master-kubelet                                           fac5a23280e6467ec95bb0237fae3ce387d04f0b   2.2.0             54d
01-worker-container-runtime                                 fac5a23280e6467ec95bb0237fae3ce387d04f0b   2.2.0             54d
01-worker-kubelet                                           fac5a23280e6467ec95bb0237fae3ce387d04f0b   2.2.0             54d
50-enable-rps                                                                                          2.2.0             5s
99-master-baa9a00d-7409-11ea-9cdc-02012c000005-registries   fac5a23280e6467ec95bb0237fae3ce387d04f0b   2.2.0             54d
99-master-ssh                                                                                          2.2.0             54d
99-worker-baaa415e-7409-11ea-9cdc-02012c000005-registries   fac5a23280e6467ec95bb0237fae3ce387d04f0b   2.2.0             54d
99-worker-ssh                                                                                          2.2.0             54d
rendered-master-782c5c9c34201d09133e3a83fa6db5bd            6e0df82cf15edec97b685e4e71f5385cebef50d2   2.2.0             54d
rendered-master-80ff9447b6f3bb017bace3eedf8e575d            fac5a23280e6467ec95bb0237fae3ce387d04f0b   2.2.0             54d
rendered-master-9d2e28683c6f08b8e3e9e7cb262abf5d            fac5a23280e6467ec95bb0237fae3ce387d04f0b   2.2.0             33d
rendered-worker-56ba8042a89644757398af2f22b39054            fac5a23280e6467ec95bb0237fae3ce387d04f0b   2.2.0             19d
...
```

Tuning recommendation #1: Receive Packet Steering (RPS), *continued*

- Check whether the MCO has been successfully applied on all matching nodes:

```
oc get machineconfigpools
```

- Sample output on our test cluster:

```
NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED
master   rendered-master-9d2e28683c6f08b8e3e9e7cb262abf5d   True      False      False
worker   rendered-worker-7c4d650eee93c040333ffe2770ce720e   False     True       False
```

Tuning recommendation #1: Receive Packet Steering (RPS), *continued*

- Be ***patient***: this process takes quite some time to complete

- You can check the current status of the nodes with the following command:

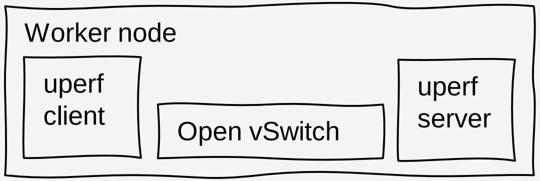
```
watch -n 5 oc get nodes
```

- Sample output on our test cluster:

```
NAME                          STATUS                          ROLES           AGE   VERSION
master-0.<our_domain_name>    Ready                           master,worker   54d   v1.14.6-152-g117ba1f
master-1.<our_domain_name>    Ready                           master,worker   54d   v1.14.6-152-g117ba1f
master-2.<our_domain_name>    Ready                           master,worker   54d   v1.14.6-152-g117ba1f
worker-0.<our_domain_name>    Ready,SchedulingDisabled        worker          54d   v1.14.6-152-g117ba1f
worker-1.<our_domain_name>    Ready                           worker          54d   v1.14.6-152-g117ba1f
worker-2.<our_domain_name>    Ready                           worker          54d   v1.14.6-152-g117ba1f
worker-3.<our_domain_name>    Ready                           worker          54d   v1.14.6-152-g117ba1f
worker-4.<our_domain_name>    Ready                           worker          54d   v1.14.6-152-g117ba1f
```

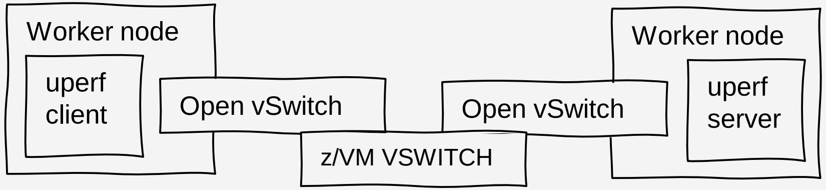
Tuning recommendation #1 (RPS): results

- **Scenario #1**: pod-to-pod, same worker



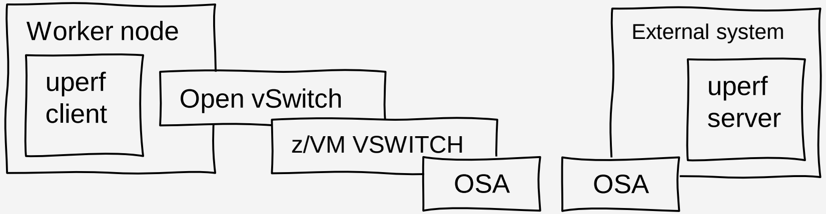
- Enabling RPS is ***not*** recommended if your OpenShift network traffic is mostly within the same worker node, since both latency and throughput ***degrade*** with this option turned on

- **Scenario #2**: pod-to-pod, cross-worker



- In this scenario, enabling RPS is ***recommended***, since it improves (a) latency ***up to 59%*** and (b) throughput ***up to 2.7x***, depending on the network connectivity and number of parallel connections

- **Scenario #3**: pod-to-external



- In this scenario, enabling RPS is ***recommended***, since it improves latency ***up to 35%***, depending on the network connectivity and number of parallel connections; throughput is either equal or higher, except for two individual measurements

Tuning recommendation #2: adding one or more static route(s)

- During our performance evaluation, we noticed that all network traffic to our external uperf target system had to ***pass through*** the load balancer due to a different IP subnet (see next page)
  – One additional network hop

- For outgoing traffic, this is ***not strictly required***
  – Might be different in your environment

- Goal: allow the traffic to flow ***directly*** to the uperf target system
  – Idea: avoid the additional hop by defining a static route

- RHEL systems: additional file in `/etc/sysconfig/network-scripts` called `route-enc9000`

```
A.B.D.0/24 dev enc9000
```

- Expectation: tuning recommendation applies to ***pod-to-external*** scenario only

Tuning recommendation #2: adding one or more static route(s), *continued*

- Before:

```
[root@worker-0 ~]# route -n
Kernel IP routing table
Destination  Gateway     Genmask         Flags   Iface
0.0.0.0      A.B.C.101   0.0.0.0         UG      enc9000
A.B.C.0      0.0.0.0     255.255.255.0   U       enc9000
A.E.0.0      0.0.0.0     255.252.0.0     U       tun0
172.x.0.0    0.0.0.0     255.255.0.0     U       tun0
```

- This is the routing table on the ***worker node***, not inside the pod

- IP address of the uperf target system: `A.B.D.254`

- **Note** that `A.B.C.101` is the Bastion server, which acts as a gateway in our test cluster

- After:

```
[root@worker-0 ~]# route -n
Kernel IP routing table
Destination  Gateway     Genmask         Flags   Iface
0.0.0.0      A.B.C.101   0.0.0.0         UG      enc9000
A.B.D.0      0.0.0.0     255.255.255.0   U       enc9000
A.B.C.0      0.0.0.0     255.255.255.0   U       enc9000
A.E.0.0      0.0.0.0     255.252.0.0     U       tun0
172.x.0.0    0.0.0.0     255.255.0.0     U       tun0
```

- **Result**: no more traffic through the Bastion server in scenario #3

- Note that ***incoming*** traffic still has to flow through the load balancer

Tuning recommendation #2: adding one or more static route(s), *continued*

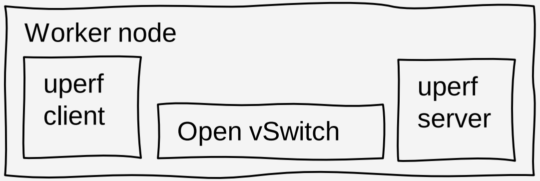
- Create a YAML file called `add-static-route.yaml` with the following contents:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 51-add-static-route
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=US-ASCII,A.B.D.0/24%20dev%20enc9000%0A
        filesystem: root
        mode: 0644
        path: /etc/sysconfig/network-scripts/route-enc9000
```

- Activate this configuration with `oc create -f add-static-route.yaml`

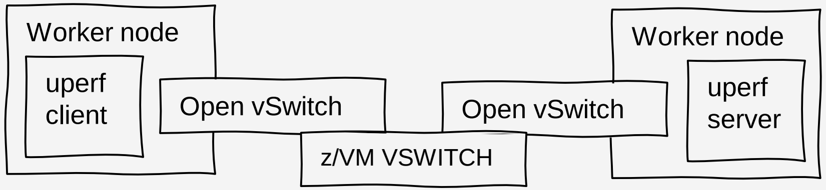
Tuning recommendation #2 (static route): results

- **Scenario #1**: pod-to-pod, same worker



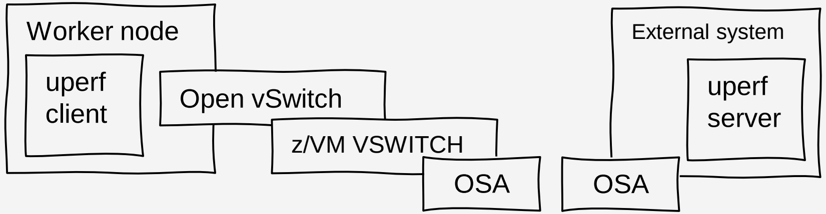
- Result: ***neutral*** – as expected, configuring a static route doesn't impact this scenario. Or in other words: it doesn't hurt, but it doesn't help either.

- **Scenario #2**: pod-to-pod, cross-worker



- Result: ***neutral*** – as expected, configuring a static route doesn't impact this scenario. Or in other words: it doesn't hurt, but it doesn't help either.

- **Scenario #3**: pod-to-external



- In this scenario, configuring a static route is ***recommended***, since it ***further*** improves (a) latency ***up to 51%*** and (b) throughput ***up to 52%***, depending on the network connectivity and number of parallel connections

Tuning recommendation #3: relocate heavy components

- In our performance analysis, we noticed that both latency and throughput were ***varying heavily***, depending on which exact worker node was selected for running the uperf tests
- Goal: achieve the ***same*** network ***performance*** on all worker nodes
  - Idea: use dedicated ***infrastructure*** nodes
- Relocating components is a ***2-step*** process:
  - First, define infrastructure nodes
  - Second, tell OpenShift to move network-heavy components out to infra nodes

Tuning recommendation #3: relocate heavy components

- Turn a ***worker*** node into an ***infra*** node with `oc edit node <name_of_the_node>` and change:

```
metadata:
  labels:
    node-role.kubernetes.io/worker: ""
```

  to:

```
metadata:
  labels:
    node-role.kubernetes.io/infra: ""
```

- You might want to ***add*** worker nodes to your cluster ***before*** converting worker nodes

Tuning recommendation #3: relocate heavy components, *continued*

- Create a YAML file with the following content:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |+
    alertmanagerMain:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    prometheusK8s:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    prometheusOperator:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    grafana:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
```

… (continued on the right) …

… (continued from the left) …

```
    k8sPrometheusAdapter:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    kubeStateMetrics:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    telemeterClient:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    openshiftStateMetrics:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
```

- All components are related to ***monitoring***
- Activate this configuration with

```
oc create -f <your_file_name_here>.yaml
```

Tuning recommendation #3: relocate heavy components, *continued*

- You can check the current ***placement*** of the corresponding pods with the following command:

```
watch -n 5 oc get pods -n openshift-monitoring -o wide
```

- Sample output on our test cluster after applying this configuration:

```
NAME                            READY   STATUS    NODE
alertmanager-main-0             5/5     Running   worker-4.<our_domain_name>
alertmanager-main-1             5/5     Running   worker-4.<our_domain_name>
alertmanager-main-2             5/5     Running   worker-4.<our_domain_name>
…
grafana-dc7ff648b-dx2rj         2/2     Running   worker-4.<our_domain_name>
…
prometheus-k8s-0                7/7     Running   worker-4.<our_domain_name>
prometheus-k8s-1                7/7     Running   worker-4.<our_domain_name>
…
```

- **Note**: some columns were cut in the above output in order to make it fit the page

Tuning recommendation #3: relocate heavy components, *continued*

- Move the ***router*** with `oc edit ingresscontroller default -n openshift-ingress-operator -o yaml` and change:

  ```
  spec:
      nodePlacement:
        nodeSelector:
          matchLabels:
            node-role.kubernetes.io/worker: ""
  ```

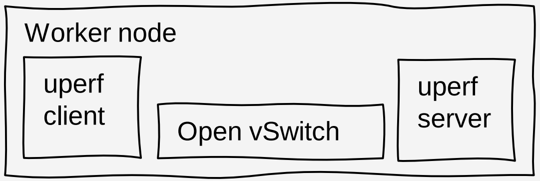
  to:

  ```
  spec:
      nodePlacement:
        nodeSelector:
          matchLabels:
            node-role.kubernetes.io/infra: ""
  ```

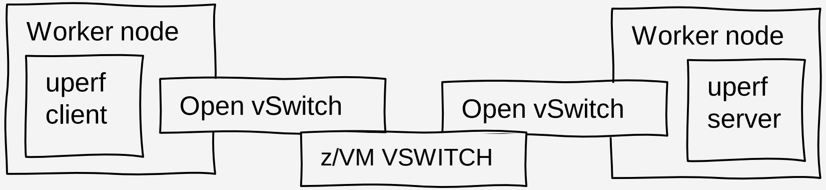
- If `nodePlacement` is missing, add it to the `spec` section

Tuning recommendation #3 (relocate components): results

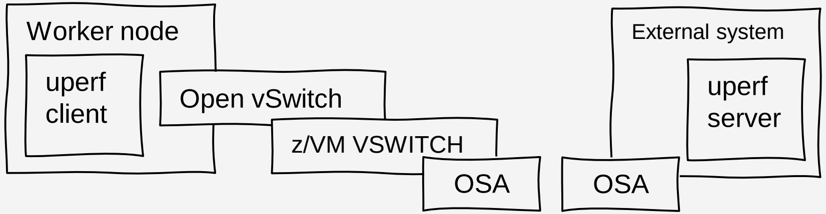
- **Scenario #1**: pod-to-pod, same worker



- **Scenario #2**: pod-to-pod, cross-worker



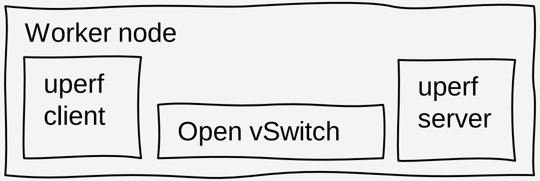
- **Scenario #3**: pod-to-external

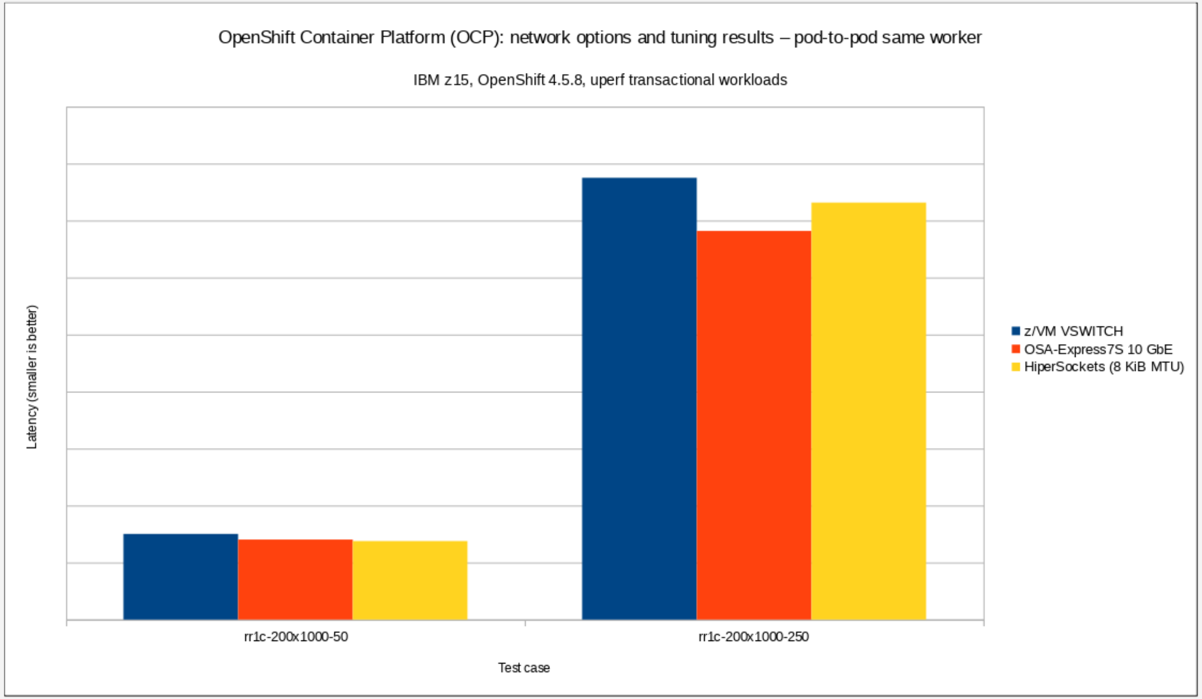


- Relocating the mentioned network-heavy components is ***recommended*** for all three scenarios

- Without this tuning recommendation, we measured latency differences between worker nodes of ***up to 87%***, despite the fact that all worker nodes were configured equal regarding CPU and memory

Scenario #1: *latency* results

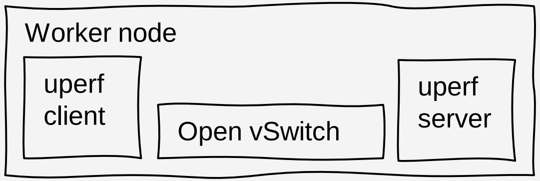


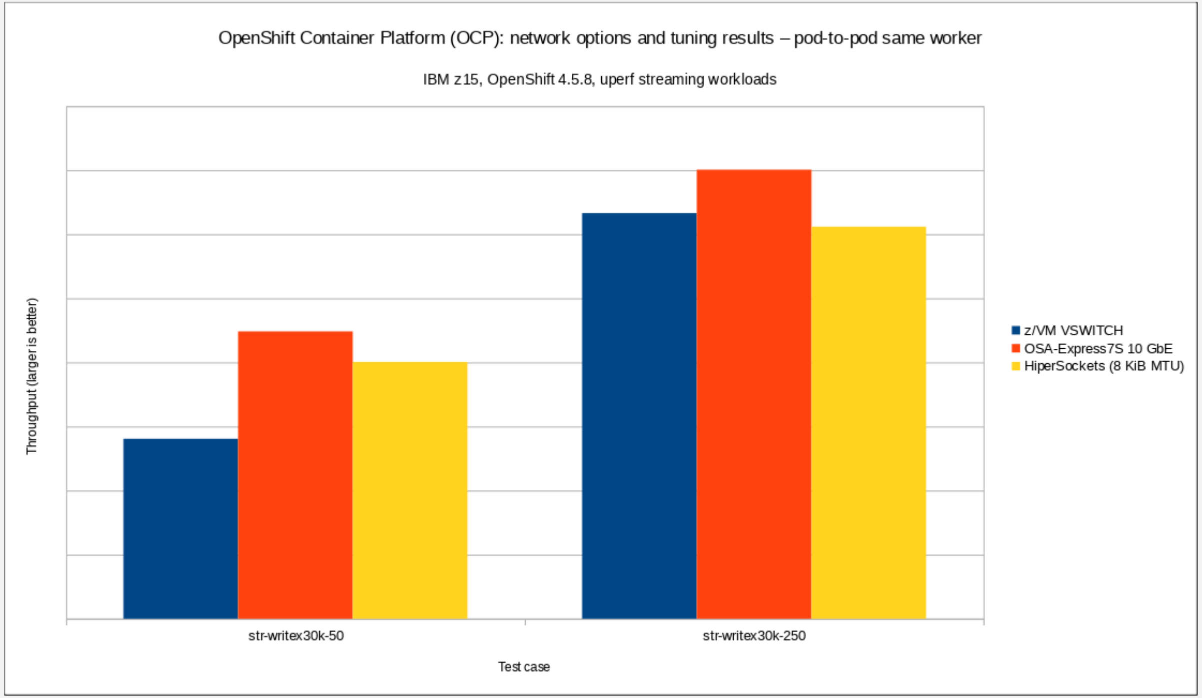


Key takeaways

- Network connectivity option – z/VM VSWITCH, OSA, or HiperSockets – only marginally impacts latency in the pod-to-pod, same worker scenario
- Enabling RPS is ***not*** recommended
- Static route doesn't impact performance in this scenario

Scenario #1: ***throughput*** results

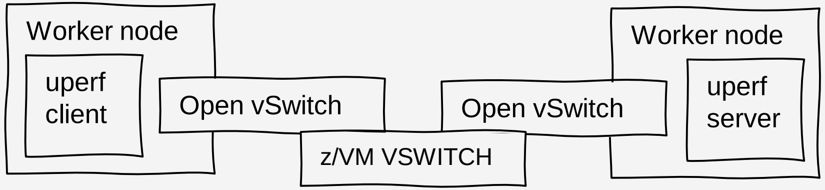


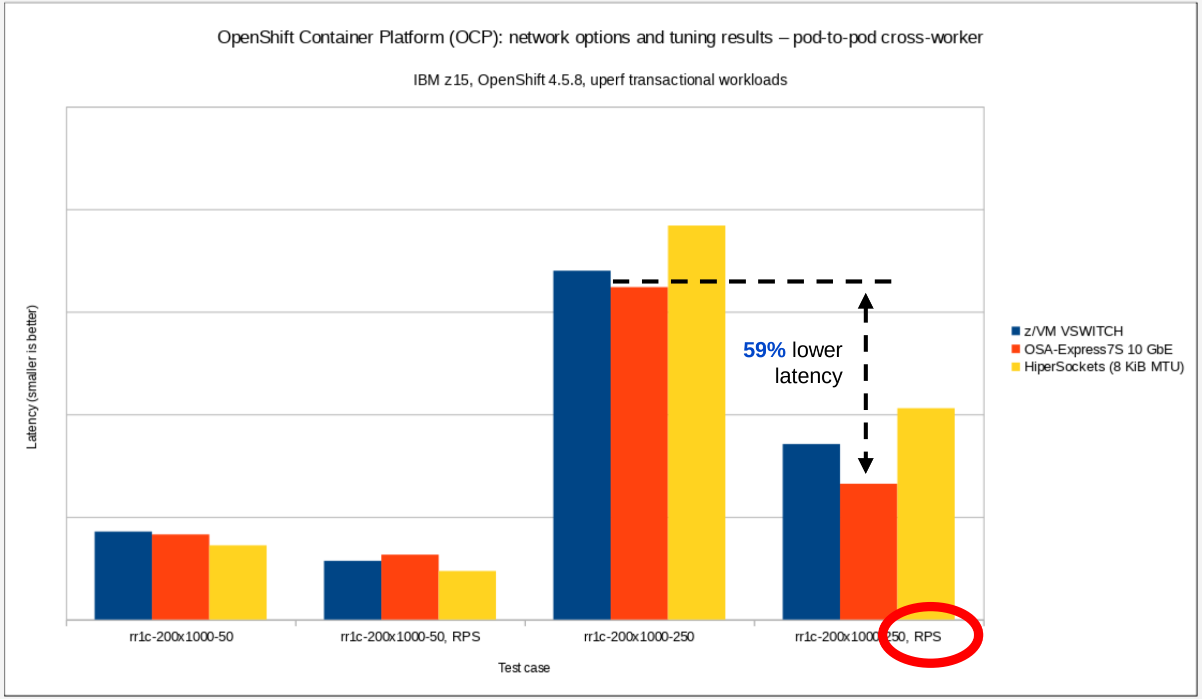


Key takeaways

- Network connectivity option only marginally impacts throughput in the pod-to-pod, same worker scenario
- Enabling RPS is ***not*** recommended
- Static route doesn't impact performance in this scenario

Scenario #2: ***latency*** results

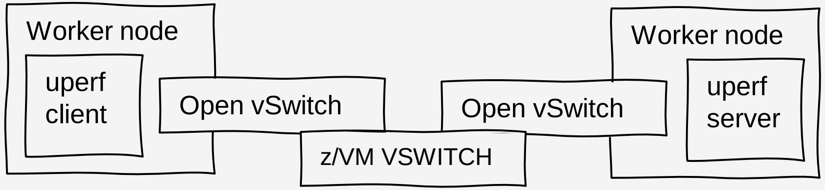


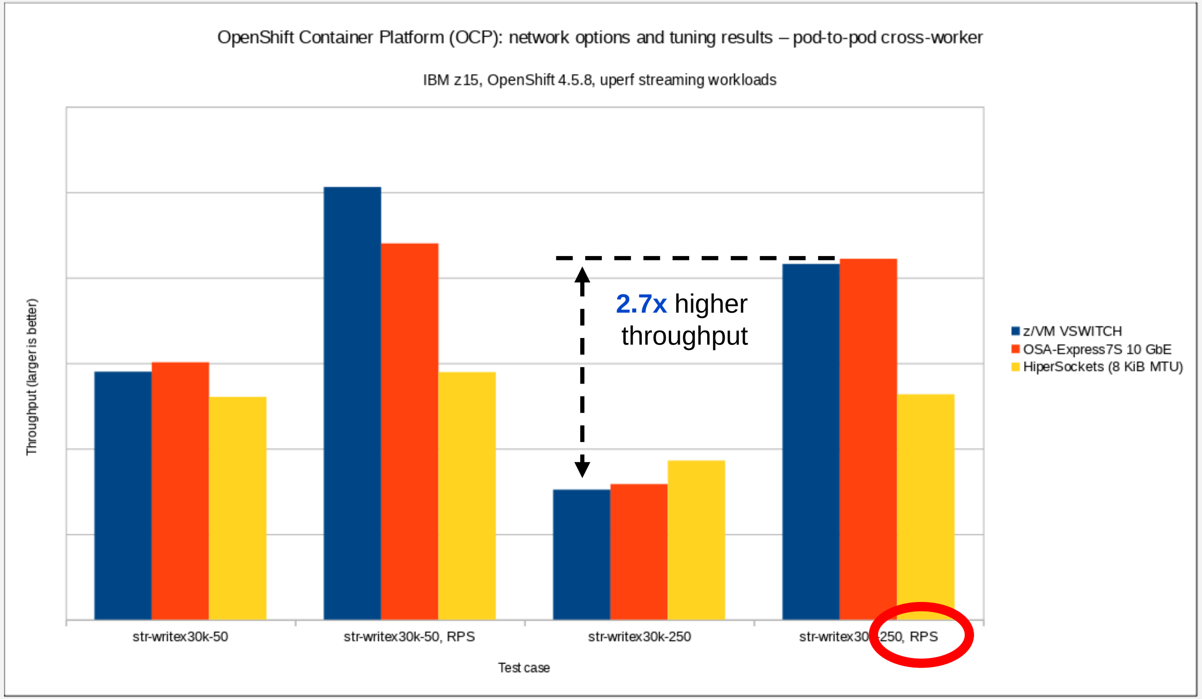


Key takeaways

- Enabling RPS is ***highly recommended*** in this scenario
- Static route has no impact
- For a high number of parallel connections (250), OSA has a ***slight*** advantage over z/VM VSWITCH and HiperSockets, but only when RPS is enabled

Scenario #2: ***throughput*** results

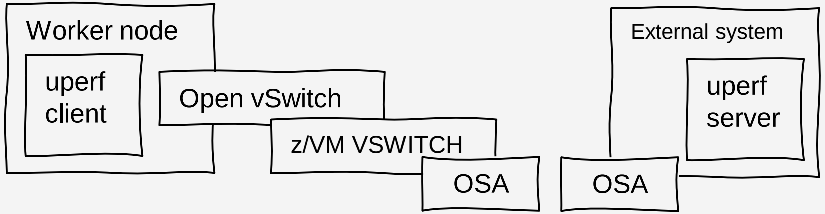


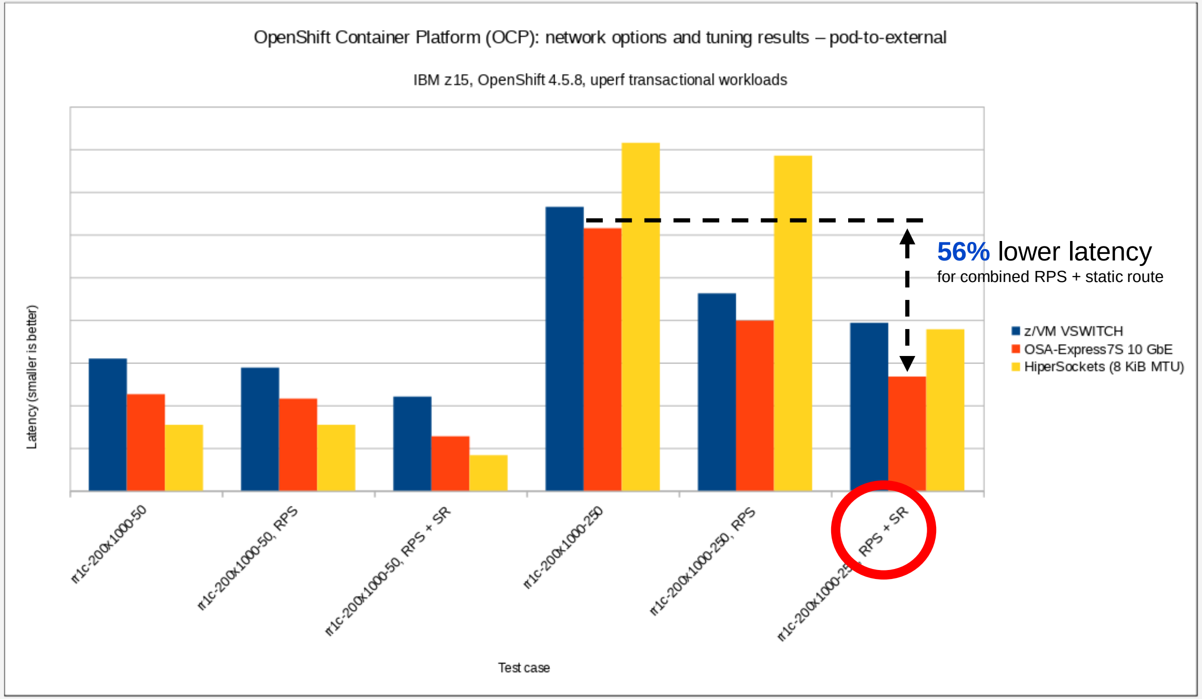


Key takeaways

- Enabling RPS is ***highly recommended*** in this scenario
- Static route has no impact
- For a medium number of parallel connections (50), z/VM VSWITCH has a ***slight*** advantage over OSA and HiperSockets, but only when RPS is enabled
- HiperSockets throughput performance can be increased significantly by adjusting ***MTU*** size

Scenario #3: ***latency*** results

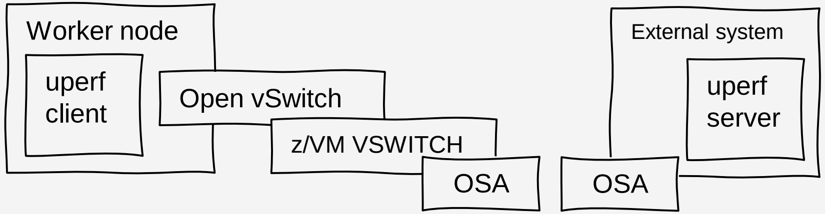


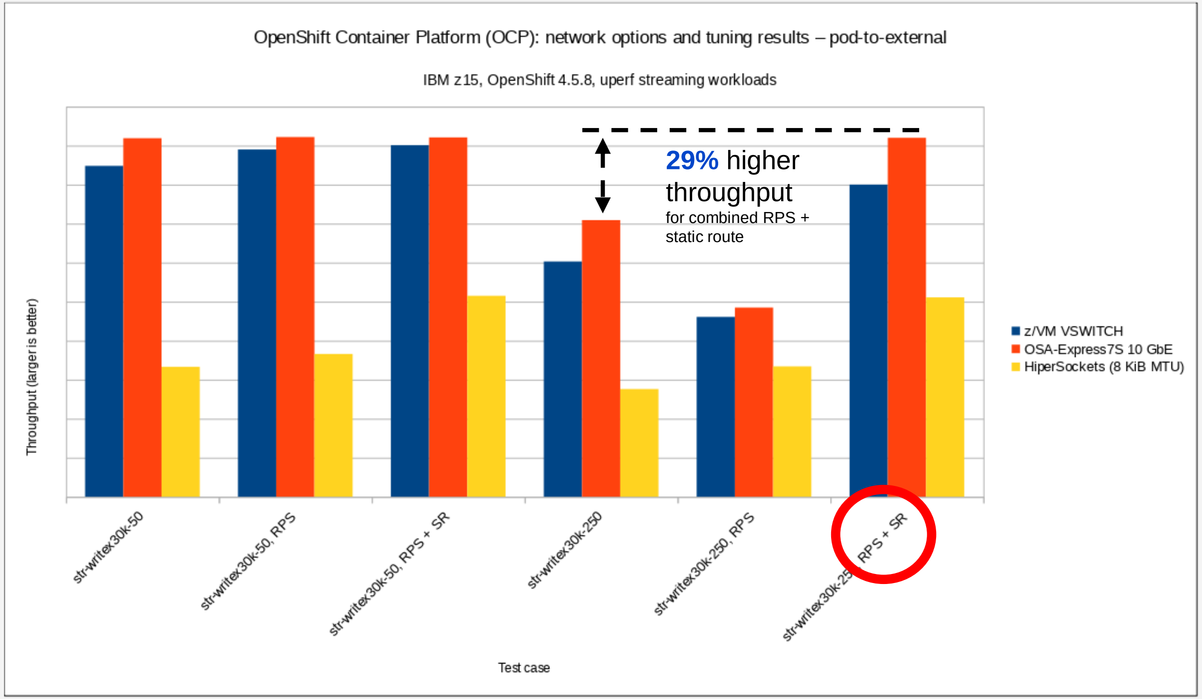


Key takeaways

- Enabling RPS and adding a static route is ***highly recommended*** in this scenario
- For a medium number of parallel connections (50), HiperSockets have a ***clear*** advantage over z/VM VSWITCH and OSA
- For a high number of parallel connections (250), OSA has a ***slight*** advantage over z/VM VSWITCH and HiperSockets

Scenario #3: ***throughput*** results





Key takeaways

- Enabling RPS and adding a static route is ***highly recommended*** in this scenario
- For a medium number of parallel connections (50), z/VM VSWITCH and OSA are ***equally well*** suited
- For a high number of parallel connections (250), OSA has a ***slight*** advantage over z/VM VSWITCH and HiperSockets
- HiperSockets throughput performance can be increased significantly by adjusting ***MTU*** size

Agenda

- Introduction
- Performance measurement and tuning approach
- Observations and recommendations
  - CPU-intensive workloads
  - Network-intensive workloads
- General tips for cloud-native applications
- Summary

General tips for cloud-native applications

- ***Stay current*** with OpenShift releases – even minor updates can improve performance ***significantly***
  – Minor updates: 4.x.y ⇨ 4.x.y+1

- When using ECKD disks for your z/VM guests, define ***HyperPAV*** aliases
  – Not using HyperPAV aliases can turn into a bottleneck for disk I/O-intensive workloads

- For ***production*** environments, use a ***hardware-based*** load balancer
  – Software-based load balancers can also work, but need much more fine-tuning

General tips for cloud-native applications, *continued*

- ***Don't*** use OpenJDK™ ***V8*** with ***HotSpot***, since it doesn't contain a *Just-In-Time* (JIT) compiler
  – Up to ***100x*** slower than a Java SDK that contains a JIT

- For best performance on Linux on IBM Z, use latest / greatest ***IBM Java V8***

- If you have a ***strict*** requirement for OpenJDK, use OpenJDK with ***OpenJ9***
  – OpenJ9 is the Open Source version of IBM Java
  – Doesn't matter whether your requirement is for V8 or V11

- If it really ***has*** to be HotSpot for some reason, then use OpenJDK ***V11***
  – On IBM Z, OpenJDK V11 with HotSpot contains a JIT compiler sponsored by SAP®

Agenda

- Introduction
- Performance measurement and tuning approach
- Observations and recommendations
  - CPU-intensive workloads
  - Network-intensive workloads
- General tips for cloud-native applications
- Summary

Summary

- Network performance is ***extremely important*** for good overall performance of an OpenShift cluster
  – Due to the nature of cloud-native applications

- There are some easy-to-implement ***tuning recommendations*** that will boost network performance
  – Receive Packet Steering, adding one or more static route(s), relocation of network-heavy components

- Before applying any of the changes, please ***double-check*** your environment for ***workload patterns***
  – Transactional vs. streaming workloads
  – Small amount vs. large amount of parallel connections

- ***Improvements*** speak for themselves
  – Up to 59% lower latency
  – Up to 2.7x higher throughput

Thank you!

Resources

- Linux on IBM Z and IBM LinuxONE
  - Official homepage: http://www.ibm.com/systems/z/os/linux
  - Documentation: http://www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_lib.html
  - Tuning hints and tips: http://www.ibm.com/support/knowledgecenter/linuxonibm/liaag/tuning/tuning.htm

- Red Hat OpenShift Container Platform Documentation
  https://docs.openshift.com/container-platform/4.5/welcome/index.html

- Red Hat OpenShift on IBM Z Installation Guide
  http://www.redbooks.ibm.com/abstracts/redp5605.html

- Relocating OpenShift infrastructure components
  https://docs.openshift.com/container-platform/4.5/machine_management/creating-infrastructure-machinesets.html

- IBM Java SDK Downloads
  https://www.ibm.com/support/pages/java-sdk-downloads

- OpenJDK 8 with OpenJ9 on AdoptOpenJDK
  https://adoptopenjdk.net/releases.html?variant=openjdk8&jvmVariant=openj9