Unleashing the Network of Red Hat OpenShift Container Platform on z/VM and RHEL KVM for IBM Z & LinuxONE

When to use Multus CNI and other tunings

Dr.-Ing. Axel Busch
axel.busch@ibm.com
Chapter Lead OpenShift Performance on IBM Z
Linux on IBM Z Performance

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a more complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*CICS®, DataPower®, DB2®, e business(logo)®, ESCON, eServer, FICON®, IBM®, IBM (logo)®, IMS, MVS, OS/390®, POWER6®, POWER6+, POWER7®, Power Architecture®, PowerVM®, PureFlex, PureSystems, S/390®, Sysplex Timer®, System p®, System p5, System x®, System z®, System z9®, System z10®, WebSphere®, X-Architecture®, z13®, z13s®, z Systems®, z9®, z/Architecture®, z/OS®, z/VM®, z/VSE®, zEnterprise®, zSeries®, IBM Z®, IBM z Systems®, IBM z13®, IBM z13s®, IBM z14®, IBM LinuxONE

The following are trademarks or registered trademarks of other companies.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Open vSwitch and OvS are trademarks of The Linux Foundation.

*** All other products may be trademarks or registered trademarks of their respective companies.**

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured Sync new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

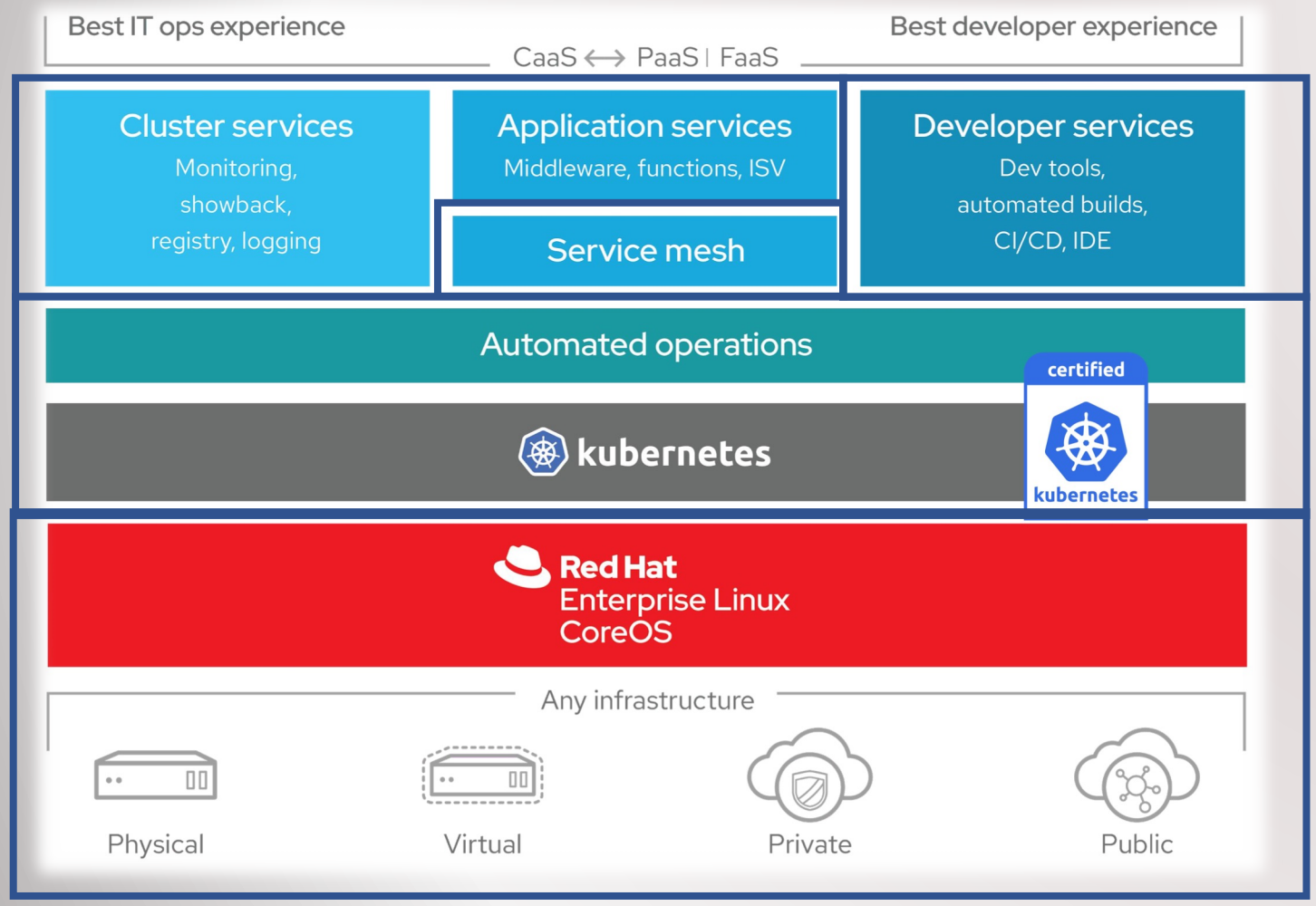
This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained Sync the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the 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.

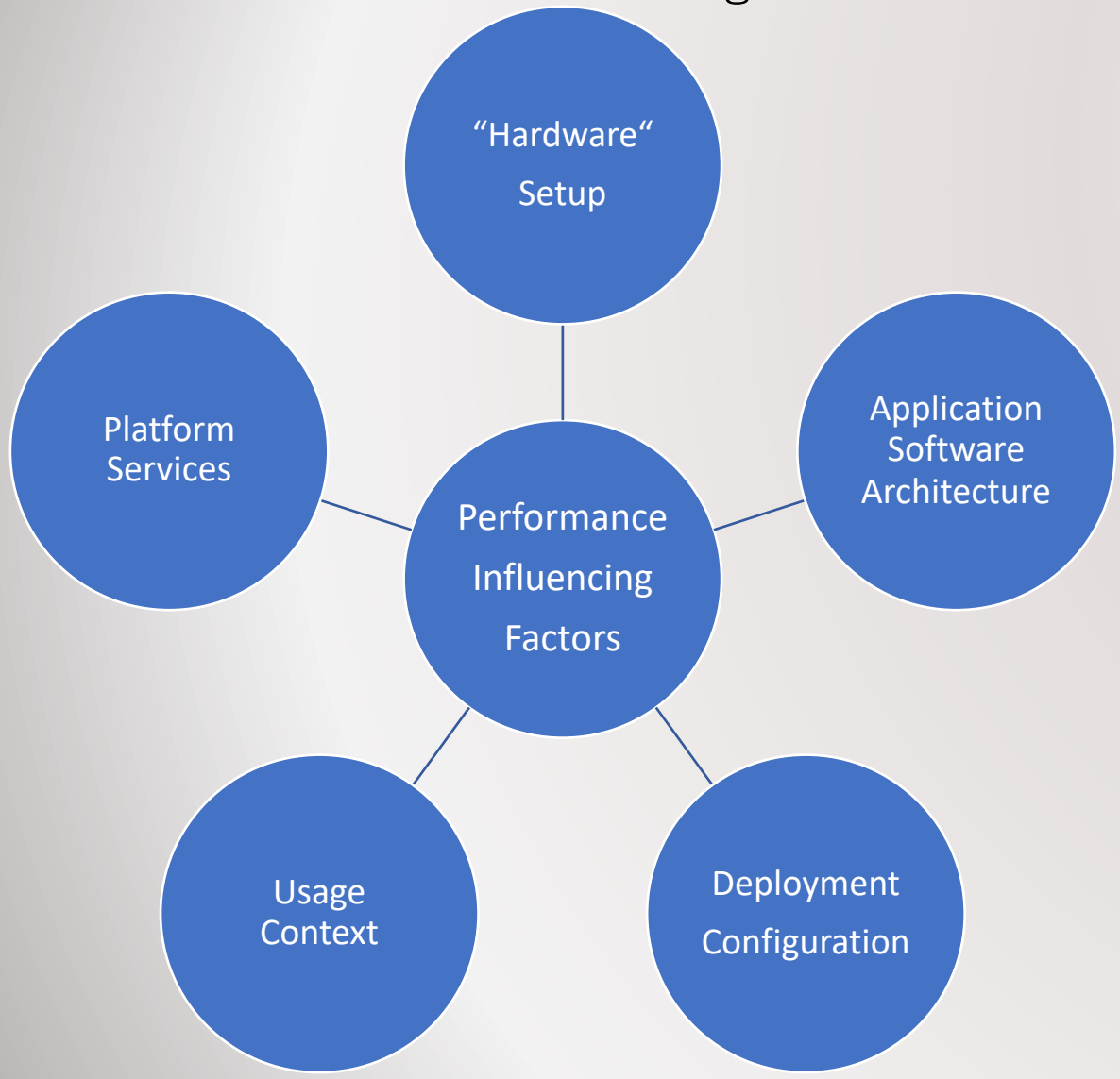
Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Red Hat OpenShift Container Platform (RHOCP) in a Nutshell



- Manage
  - resources, .e.g. public/private cloud
  - infrastructure, e.g. network
  - operating system, e.g. Linux performance settings
- Development
  - of applications using CI/CD pipeline
  - … in different programming languages
  - … and code repositories
- Deployment
  - of applications in containers
  - … to be scaled
  - … and made reliable
- Operation
  - Monitoring, logging
  - Middleware, operators
  - Network topology

Performance and its influencing factors



- CPU/memory/.. resources
- Hypervisor, i.e. z/VM, KVM,…

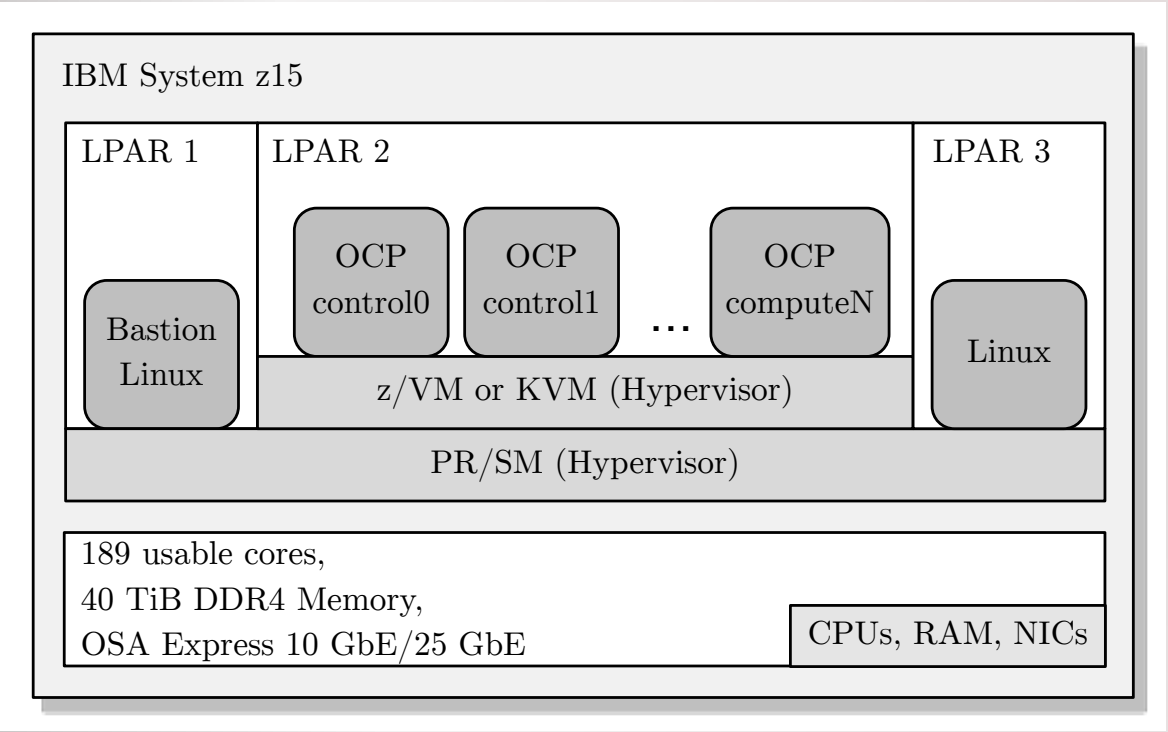
- Monolythic Application
- Service oriented
- Microservices

- Trade-off decision for deployment
- High locality vs. high distribution

- Workload pattern
- # concurrent users
- Data amount

- Kubernetes Layers/Operators
- Software defined network
- Prometheus
- etcd,…

General system setup: IBM z15

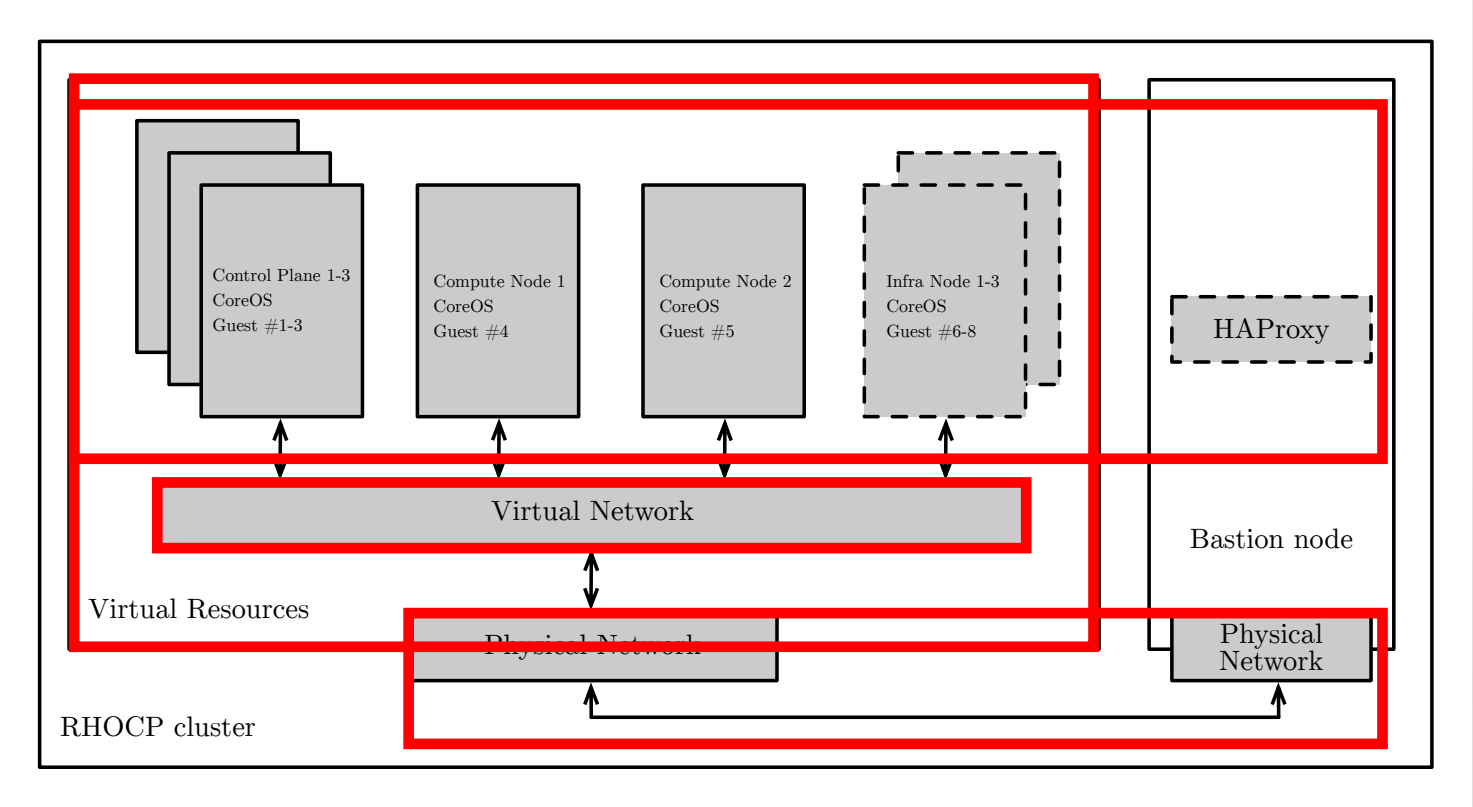


RHOCP LPAR resource configuration
- 16 IFLs + SMT2 each
- 200 GB memory each
- Dedicated OSA Express 7s 10 GBit/s to each LPAR

RHOCP configuration
- 3 control planes with 4 vCores, 32 GiB memory each
- 4 compute nodes with 4 vCores, 32 GiB memory each
- 2 infra nodes with 4 vCores, 32 GiB memory each
- bastion with 16 vCores, 16 GB, dedicated OSA Express 7s (LPAR 1)

External LPAR 3
- RHEL 8.x Linux
- 32 vCores, 32 GiB memory, dedicated OSA Express 7s

RHOCP Cluster architecture general



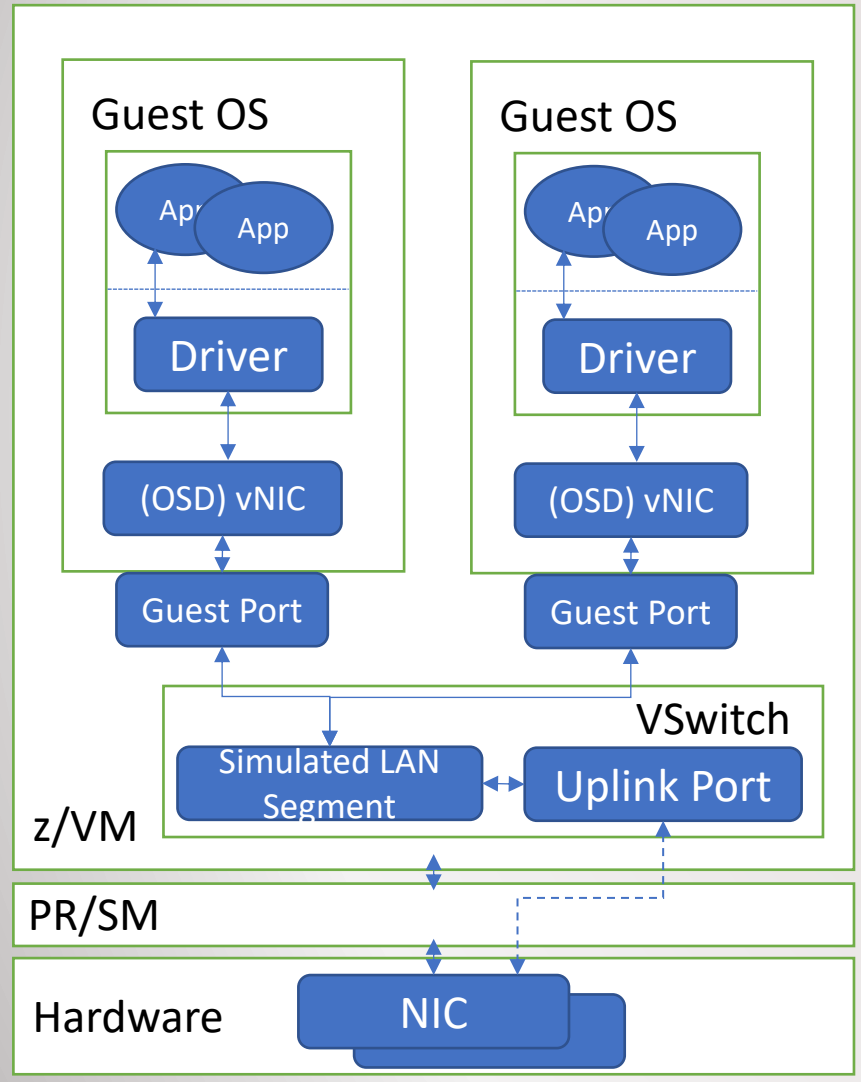
OCP nodes
- Control planes
- Compute nodes
- Infrastructure nodes
- Bastion node (for e.g. perimeter network)
- All infrastructure related pods (i.e. router, prom,...) moved to infrastructure nodes

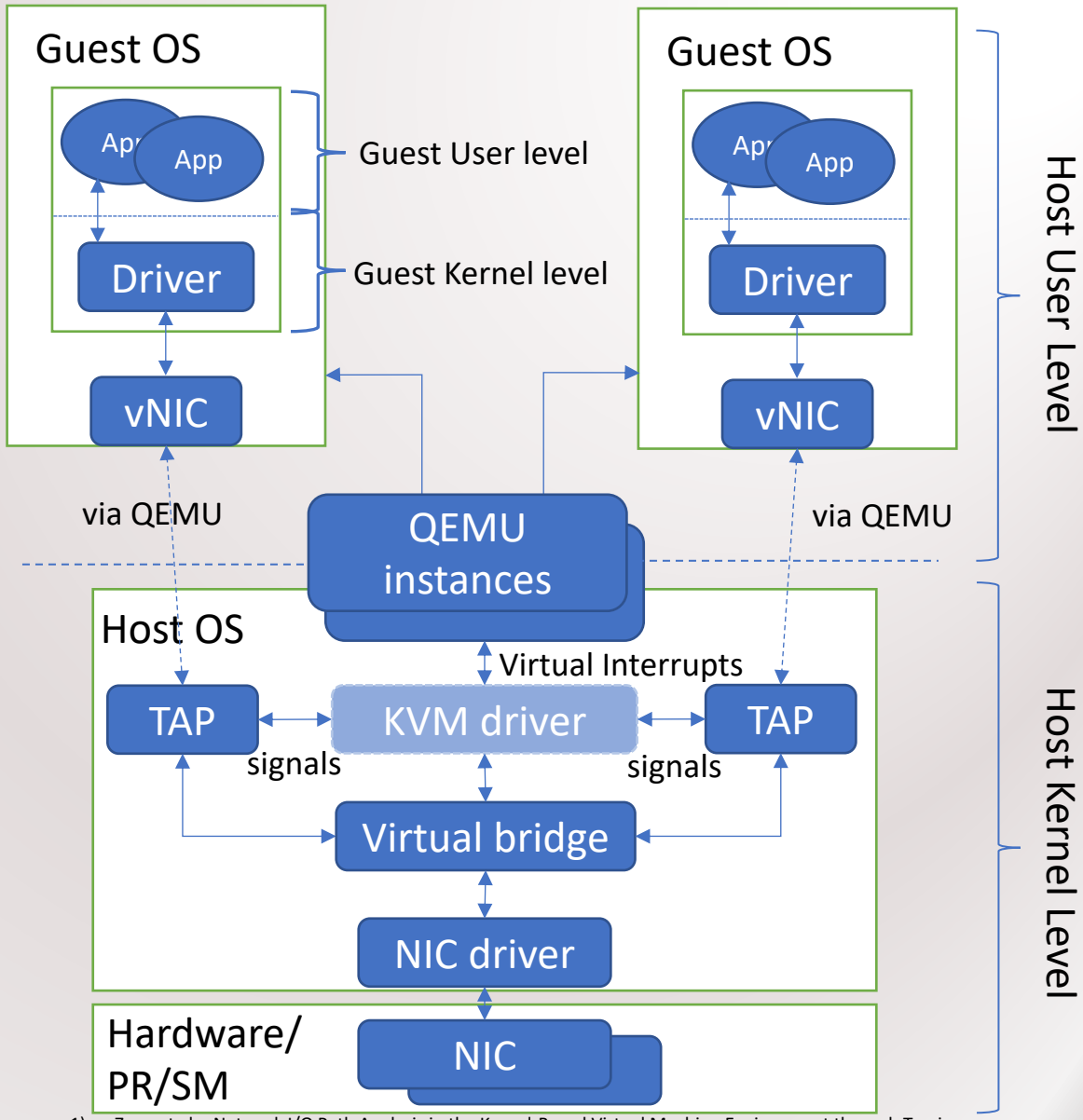
Virtual resources
- KVM on IBM Z & z/VM hypervisor
- z/VM VSWITCH
- Tap + bridge

Virtual OCP network
- SDN Provider: OpenShift-SDN (replaced by OVN-kubernetes)
- CNI Plugins

Physical Network
- OSA Express 7s NICs

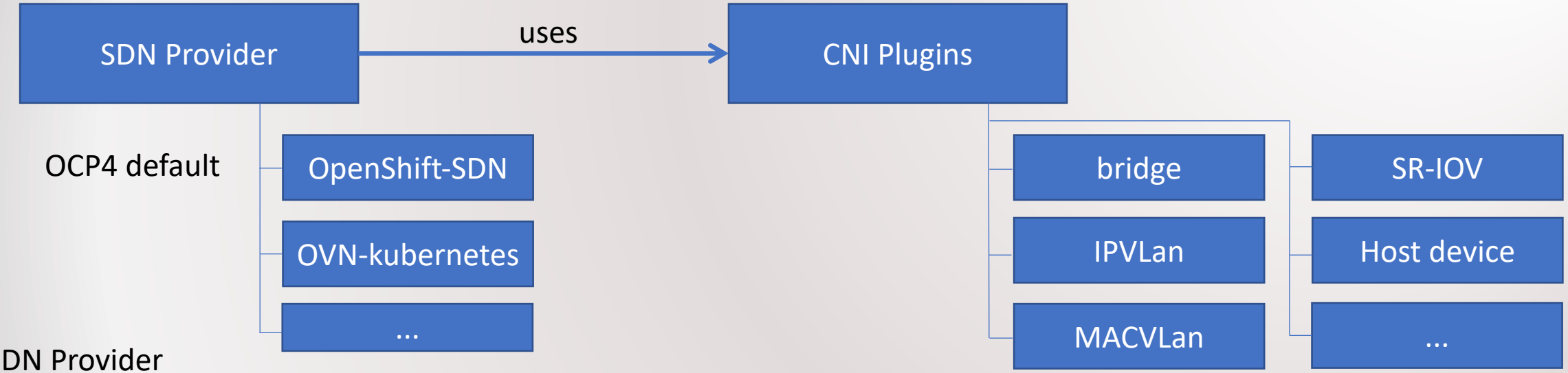
Network: z/VM & VSWITCH vs. KVM & Tap+bridge





1) Zeng et al. - Network I/O Path Analysis in the Kernel-Based Virtual Machine Environment through Tracing
2) Ren et al. - Shared-Memory Optimizations for Inter-Virtual-Machine Communication

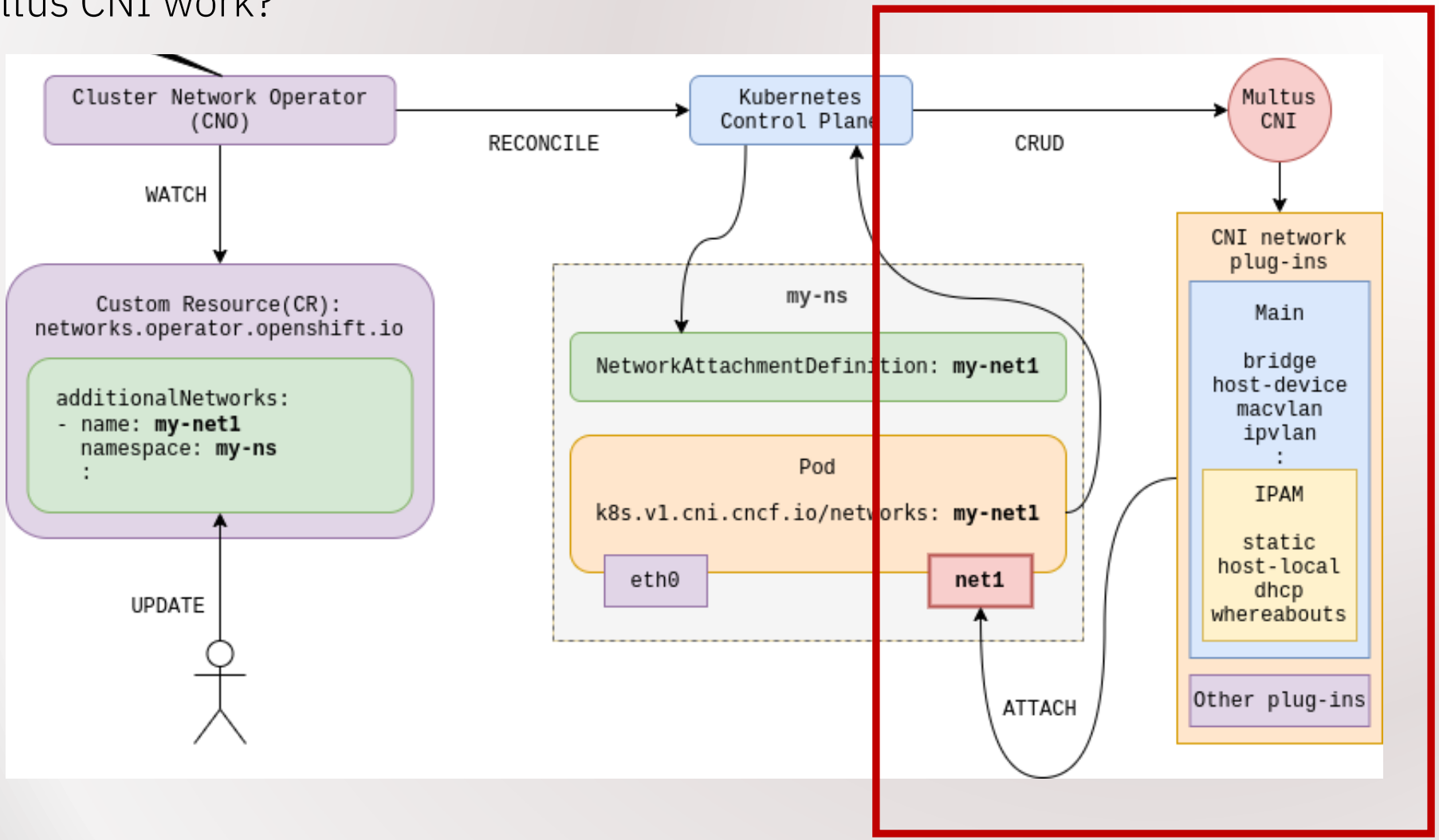
Software defined network (SDN) provider: High level overview



SDN Provider
- Attaches virtual interfaces (provided by CNI plugins) to containers
- Routes traffic between virtual interfaces by tables
- Defines higher-level functions for packet processing (e.g. firewall)

CNI (Plugins)
- Container virtual interface is a specification (i.e. a contract)
- Defines container‘s network connectivity
- Defines how to allocate and remove resources required for the connectivity
- Plugins implement CNI
- Can be attached by the SDN provider to the containers

How does Multus CNI work?



https://cloud.redhat.com/blog/using-the-multus-cni-in-openshift

CNI Plugins: VXLAN vs. IPVLan vs. host-device

OpenShift-SDN+VXLAN

\+
- Default configuration
- Using *Multus* additional interfaces can be attached to pods
- Fast in co-located scenarios with large packet sizes

−
- Slower and lower efficiency in several scenarios especially for external connections

IPVLan

\+
- Good trade-off between performance and functionality
- Layer2 mode shows best performance in our scenarios

−
- Higher configuration effort (Multus CNI required)
- Latency and throughput slower in co-located scenario

host-device

\+
- Highest performance and efficiency for external scenarios
- Easy to configure for single pods such as a database pod

−
- Only one instance per node can use host device
- For use with Multus higher authority needed
- Might bypass some networking stack and features (NetworkingPolicy object rules)

Scenario based reconfiguration of cluster to get best performance

Network setting: Receive Flow Steering (RFS)

- RFS tunes the network performance in terms of throughput and response time
- Is a **trade-off between latency/throughput and CPU usage**
- Depending on the scenario RFS can
  - improve latency, throughput and efficiency
  - improve latency and throughput but decrease efficiency
- Can RFS considered harmful?
  - **Usually RFS has positive effects on latency, and throughput**
  - In resource contention scenarios (e.g. too few CPU cores, NIC saturation) RFS can lead to throughput degradation

If you have **too high over-commitment RFS can increase the steal-time** even more what can end up in lower performance

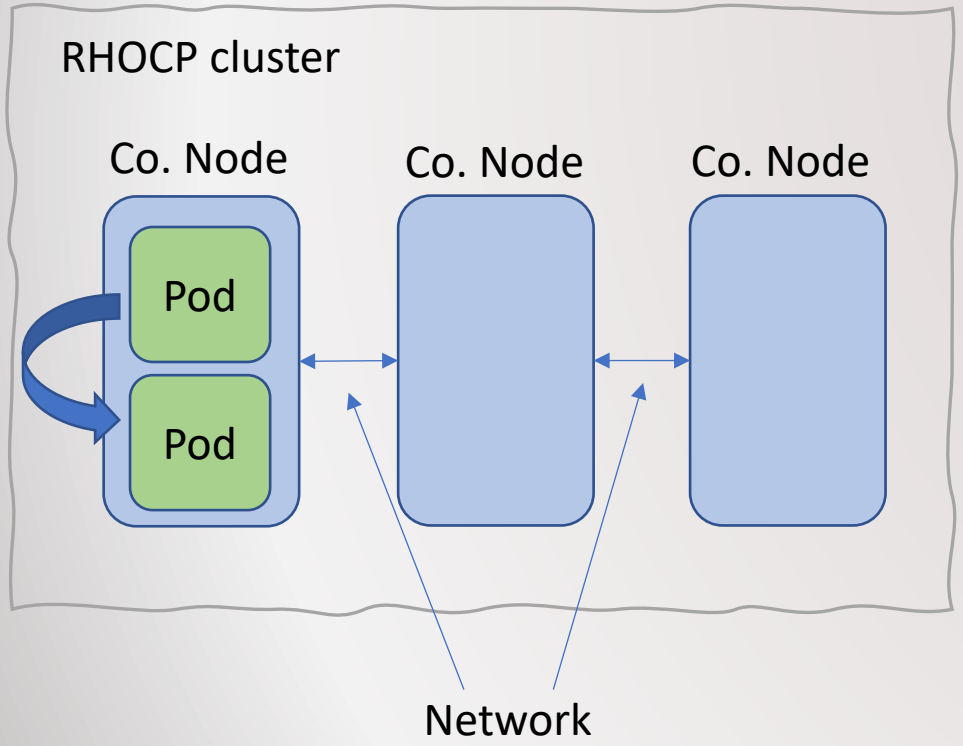
How is RFS working?

- RFS checks if packet processing is running on CPU of destination thread
- If not, the table is updated and packet processing is performed on the target core
- This uses positive cache effects more efficient
- RFS can be used with Openshift-SDN+VXLAN as well as CNI Plugins

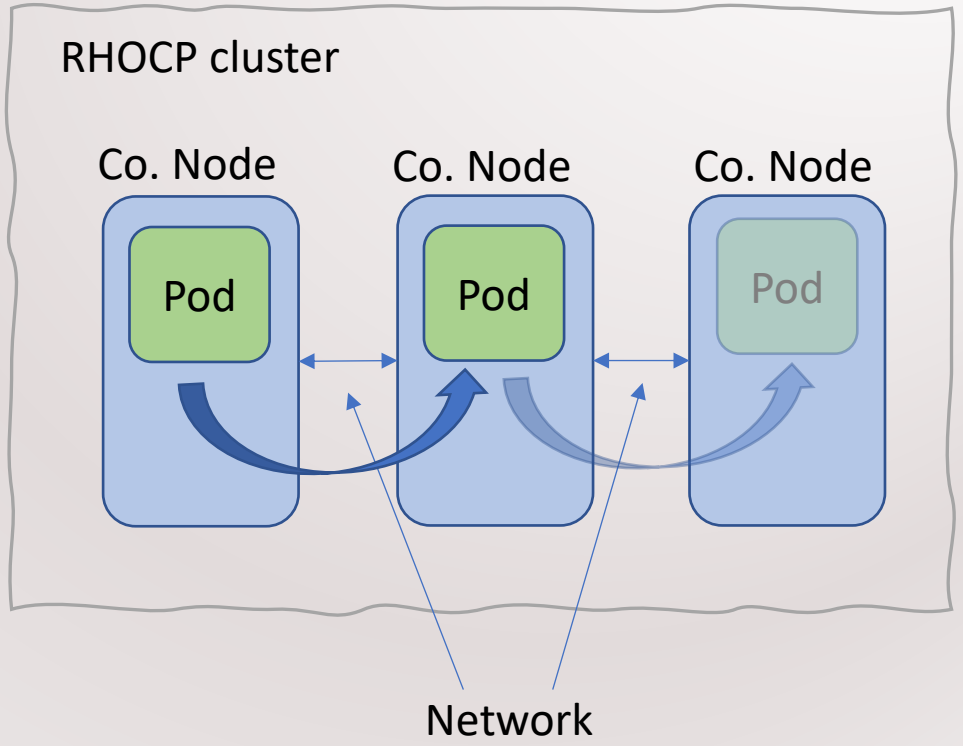
Deployment strategy: same node vs. cross node deployment

- Same node: Pods are located on the same physical or virtual machine (such as an OCP compute node)
- Cross node: Pods are spread over one or several physical or virtual machines

Same node deployment



Cross Node deployment

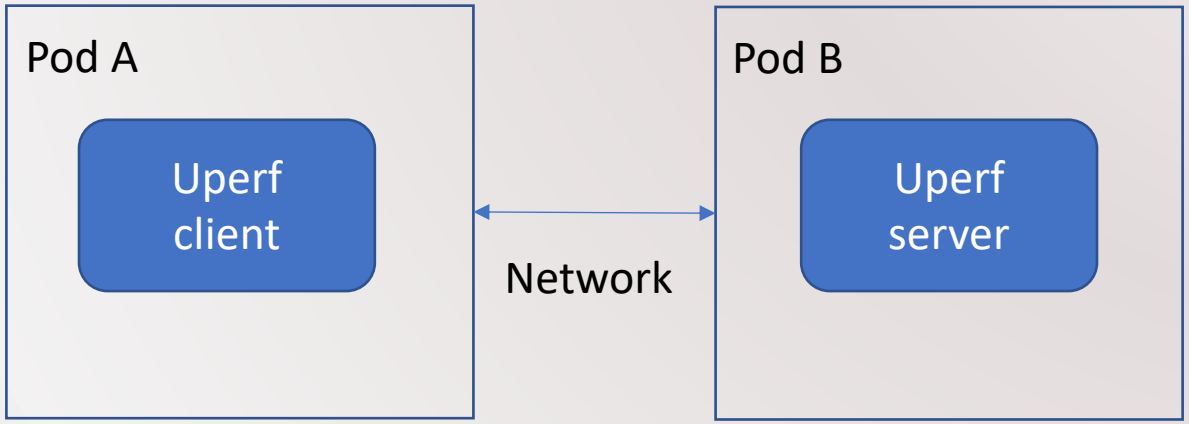


- On z/VM and KVM on IBM Z compute nodes were installed on same LPAR and CEC
- HA considerations (several LPARs, cross CEC setups) might show different results

(General) network performance tuning ideas

How to Benchmark the Network: uperf & workloads

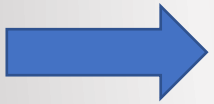
Uperf: A network (micro) benchmark
- Two sets of workloads
  - **Request Response** (latency)
  - **Streaming** (throughput)
- Several numbers of simultaneous **connections** (1-50-250)
- Different **request sizes** (1x1-200x1000-200x30000 B)



Results:
- Latency in us/ms
- Throughput in MiB/s
- CPU load, i.e. user/system/steal
- Efficiency, i.e. throughput/CPU load

Preliminary Discussions

- Controlled analysis environment
  - Dedicated IFLs (no over-commitment)
  - No other LPARs running in parallel
  - Dedicated NICs (OSA Express 7s)
  - Workload has no external dependencies

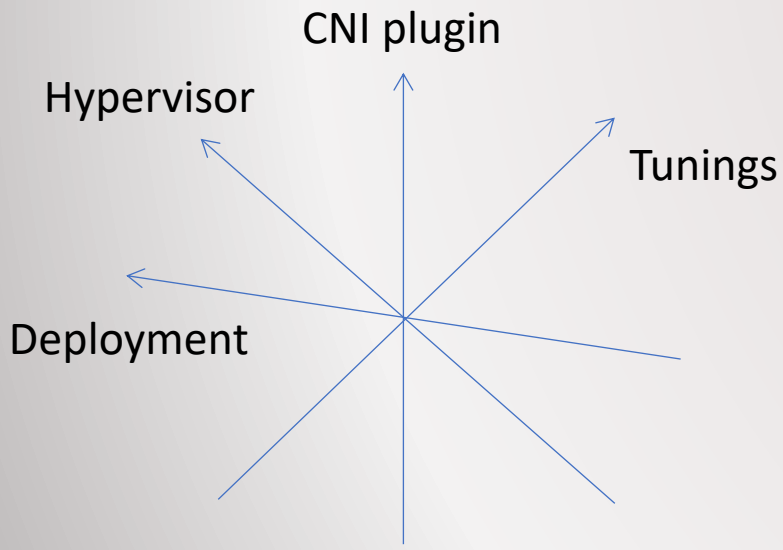


No trade-off decisions made in terms of cost, availability,... **focus on performance only**

- Technology comparison
  - Comparison of technology does not mean one is good and one is bad
  - It just means under certain conditions one technology can outperform the other
- Study stats
  - 720 measurement series
  - 60 hours measurement time
  - 10 GB resulting data to be analysed

General tuning recommendations and decision space

- **General recommendations are hard**
- The more general, the less precise - **use scenario dependant tuning hints**
- This slide deck focusses on **networking performance**
- For optimal configuration **trade-off decisions** required
- Trade off decisions can be derived from decision space:



- Hypervisor: z/VM vs. KVM on IBM Z
- Deployment: local deployment vs. distributed deployment
- Network tuning: Receive flow steering on/off
- CNI Plugins
  - Default-SDN
  - IPVLan
  - Host device

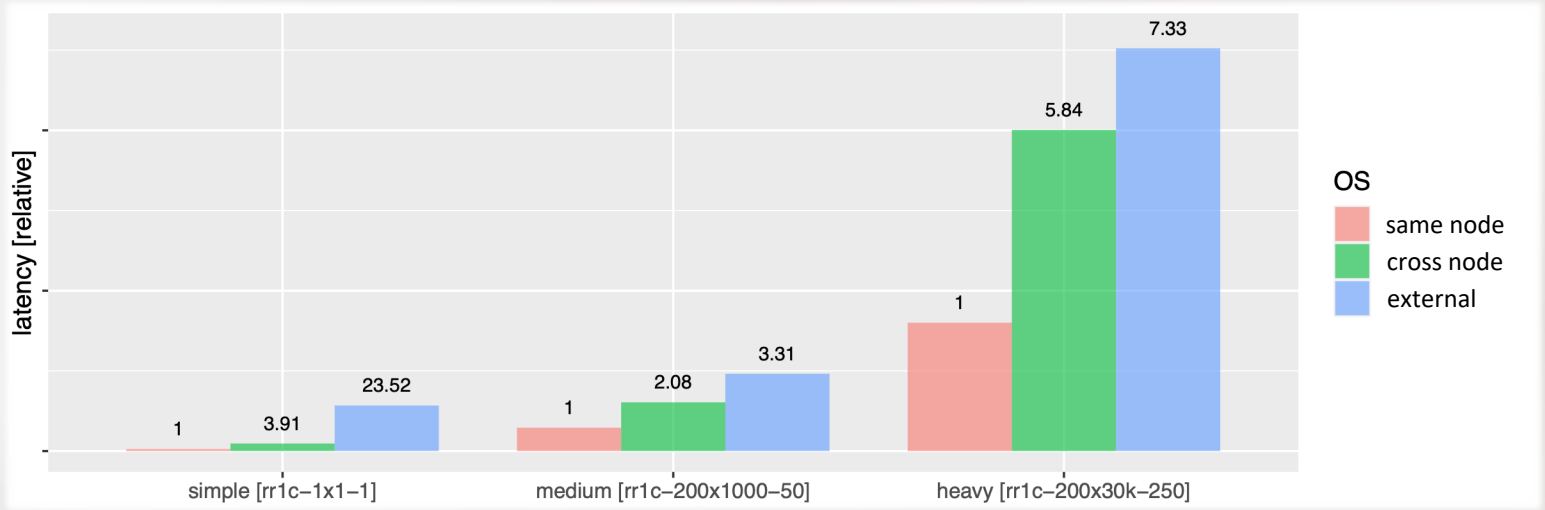
Further influencing factors:

- Number of network connections
- Network packet sizes
- Workload pattern: Request/response vs. streaming workload
- CPU over-provisioning

General (architecture) tuning ideas: Deployment

1. Deploy locally

- **Deploy** services with frequent interaction as **local as possible**
- Communication **between workers** is more **expensive**
- Communicate to external systems shows longest response times (with Openshift-SDN and VXLAN)
- **Trade-off between CPU resources, availability and scalability features required**



General (architecture) tuning ideas: Avoid over-provisioning

2. Avoid too much over-provisioning

- Over-provisioning can lead to unstable cluster
  - etcd leader changes and leader voting required
  - Consumes even more CPU
- Cluster operators working in the background
- KVM and z/VM **have to do a lot more than with usual Linux installations**
- **Steal time can go up, wastes CPU** and **prevents workload** from beeing scheduled
- **Network performance can be degraded** because z/VM VSWITCH needs resources as well
  - If VSWITCH is slow the applications waiting for network I/O might slow down
  - Ripple effects through the architecture (affecting other pods) is possible

General (architecture) tuning ideas: Avoid background tasks

3. Avoid too much background noise

- Use **LPAR dedicated to RHOCP**
- **No other services** should run on the same LPAR
- Can degrade performance of your RHOCP significantly
- RHOCP operators doing quite a lot in the background
- Might lead to concurrency situation between
  - Hypervisor scheduling
  - CPU resources
  - Networking resources
  - Disk I/O resources

General (architecture) tuning ideas: Apply tunings to your cluster

1. Use **infrastructure** nodes
2. Enable **RFS** in your cluster (more infos later in this slides)
3. Use **HPAV devices** if you use ECKD
4. **Tune your hypervisors**
    1. Tune LPAR weights
    2. Adjust sched_migration_cost_ns for KVM on IBM Z
    3. Tune z/VM CPU share

More information to be found:

- General performance overview
- OCP on Z - Performance, Quality & Best Practices
- How to setup infra nodes
- How to setup PAV devices

Hypervisor performance and efficiency: z/VM vs. KVM on IBM Z

- Overall performance / efficiency[1]
- No further decisions considered (such as deployment, workload, etc.)
- Default-SDN used
- RFS tuning on

	#cases
z/VM vs. KVM on IBM Z “overall“ performance & efficiency[2]:	7/15

- Workload pattern:

z/VM vs. KVM on IBM Z request/response (rr) performance & efficiency[2]:	6/15
z/VM vs. KVM on IBM Z streaming performance & efficiency[2]:	2/15
z/VM vs. KVM on IBM Z – rr many external connections efficiency[2]:	13/15
z/VM vs. KVM on IBM Z – rr many external connections performance[2]:	8/15

- KVM on IBM Z with higher steady state load might reduce efficiency especially with many workers beeing smaller sized
- KVM on IBM Z slower within worker internal communication (see next slide)

[1]efficiency excludes higher steady state load of the cluster. Focus on networking efficiency such as throughput per core and latency efficiency only.
[2]the bars represent how often one of the technologies/settings have shown at least 10% better performance compared to the other.

Deployment strategy: same node & cross node deployment – z/VM vs. KVM on IBM Z

- Default-SDN used
- RFS tuning on
- **Same node deployment**:

	#cases
z/VM vs. KVM on IBM Z request/response performance & efficiency:	13/15
z/VM vs. KVM on IBM Z streaming performance & efficiency:	10/15

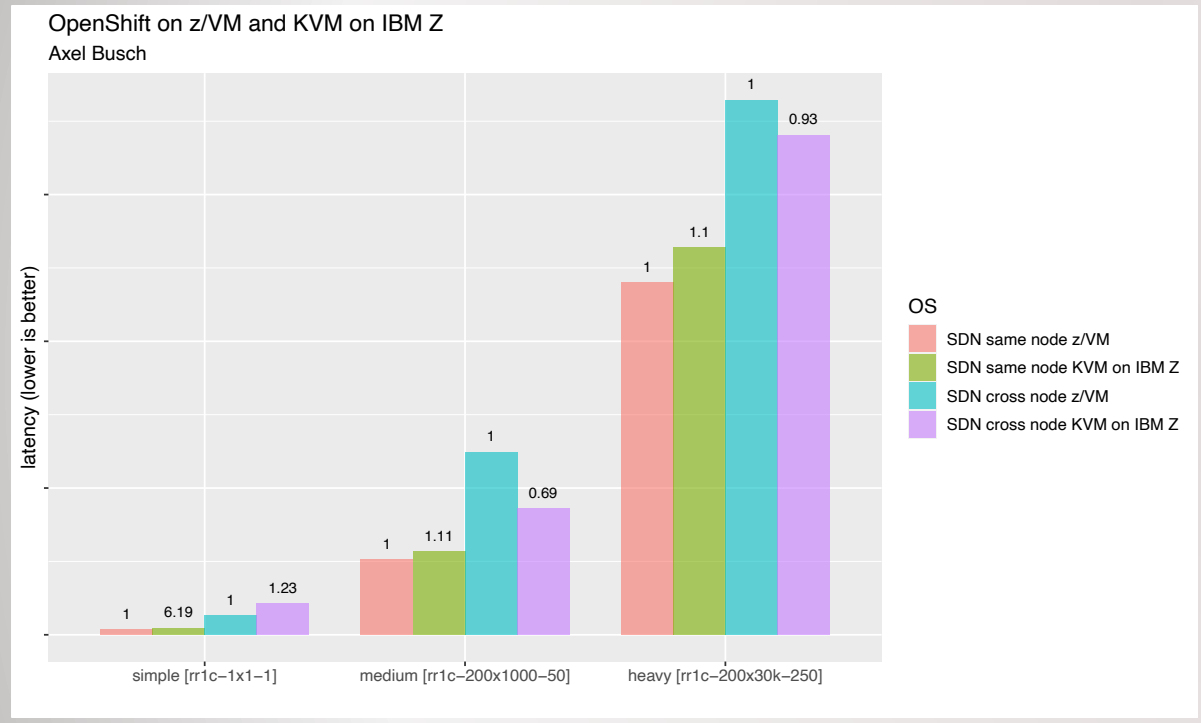
- **Cross node** deployment (i.e. client/server spread over several nodes):

z/VM vs. KVM on IBM Z request/response performance & efficiency:	7/15
z/VM vs. KVM on IBM Z streaming performance & efficiency:	3/15

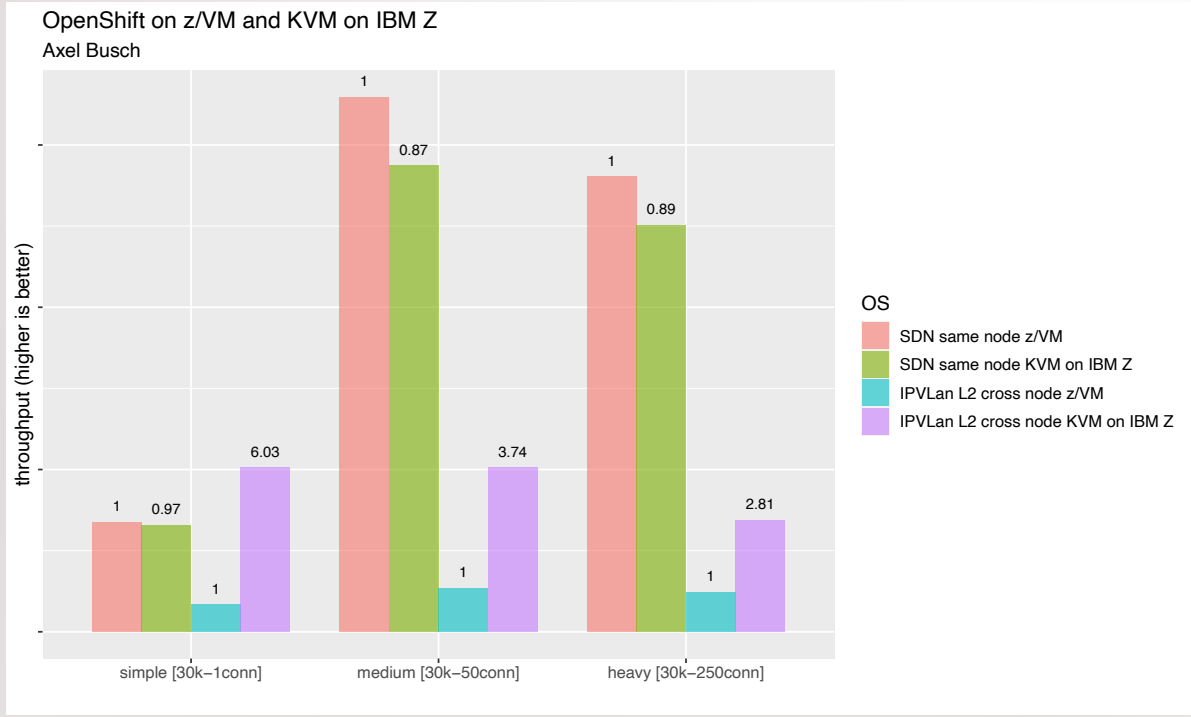
- No winner, more of a situational pro and con tradeoff
- Depending on your **deployment strategy** your decision might be in favor of z/VM or KVM on IBM Z

Let's see some values: z/VM vs. KVM on IBM Z – latency and throughput

latency

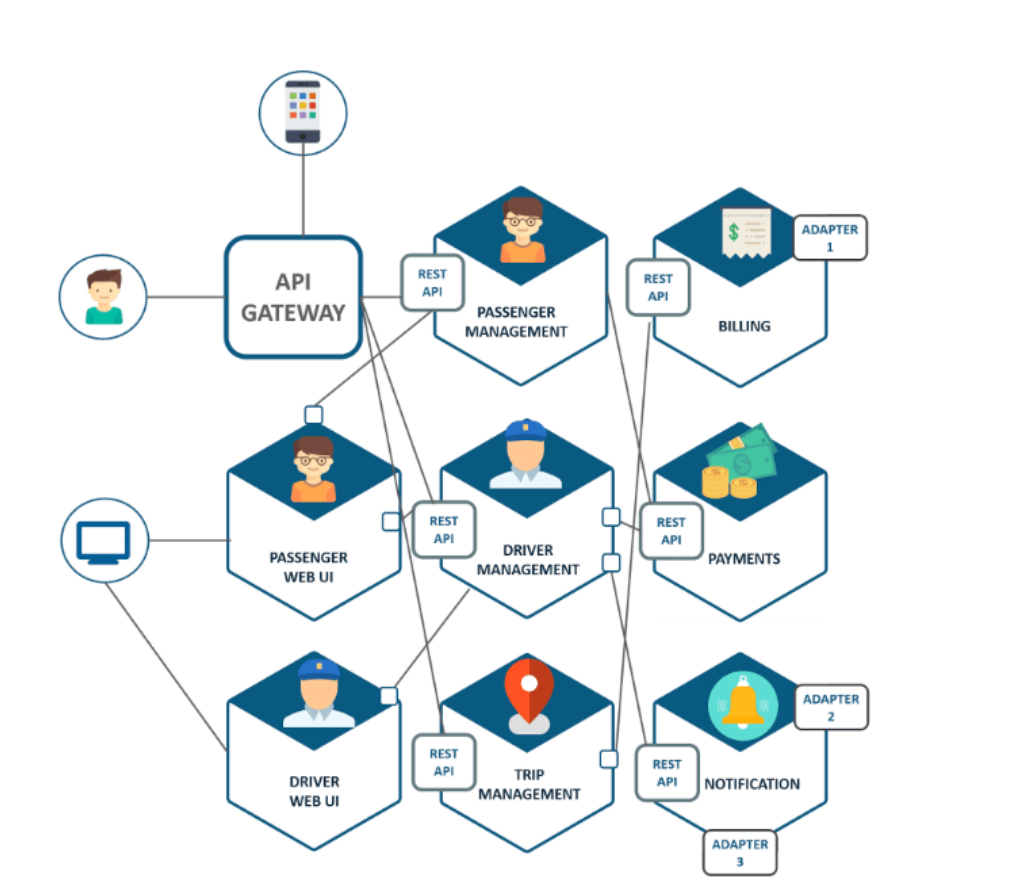


throughput



- When it comes to latency the numbers might not differ a lot
- Relevant if maximum performance should be obtained or long service chains lead to long service response times
- However, **latency can add up quickly** and must be multiplied by the **number of interactions** between pods

Example: Uber passenger Web UI

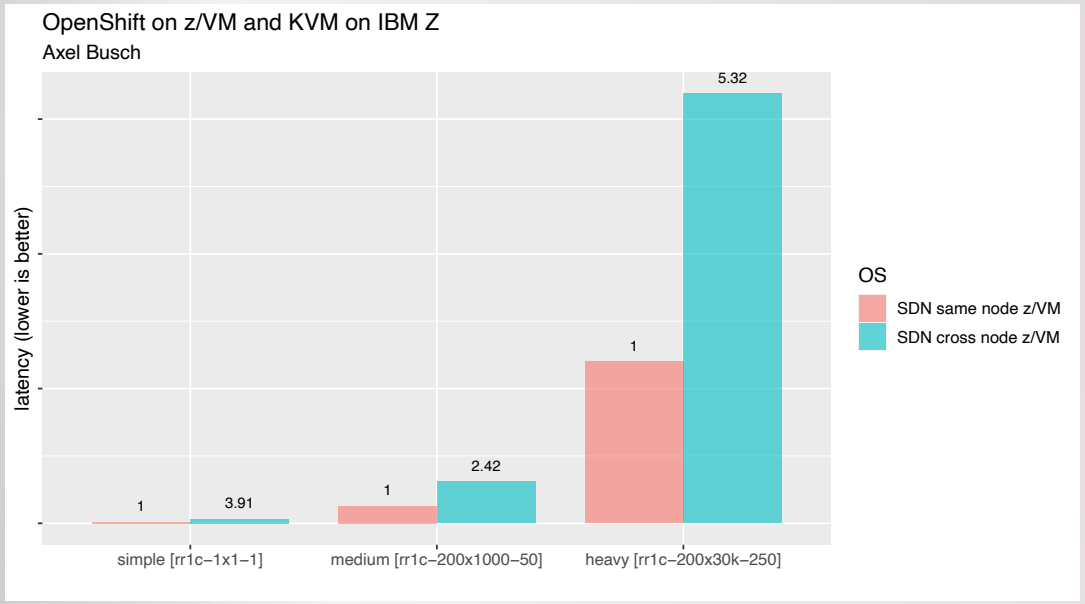


https://blog.dreamfactory.com/microservices-examples/

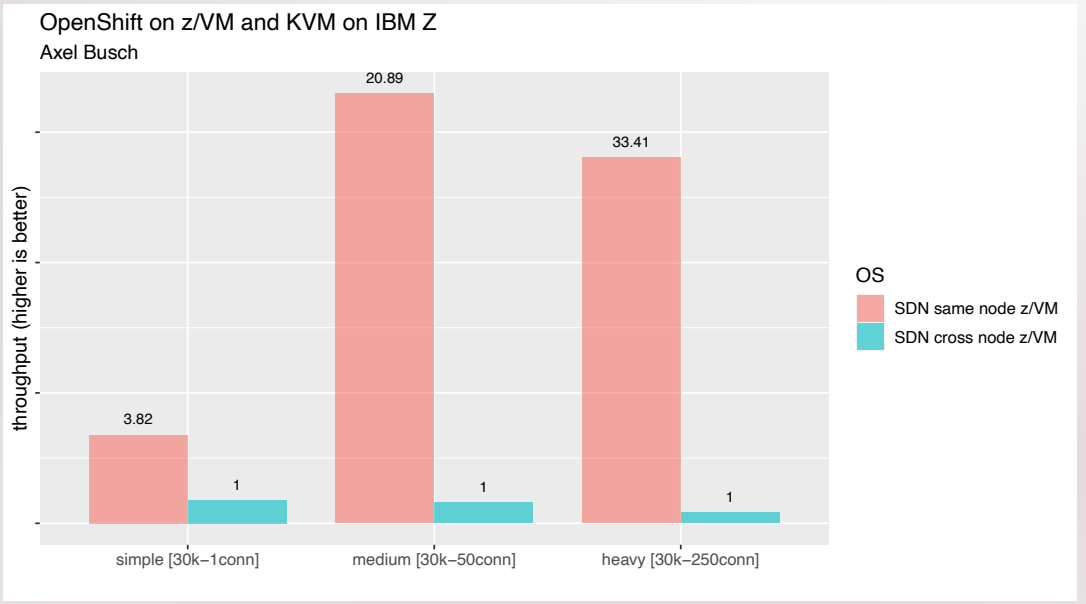
Let's see some values: same node vs. cross node deployment - latency & throughput

- Default-SDN
- No Multus CNI Plugin used
- RFS off
- **Focus on deployment**

latency



throughput

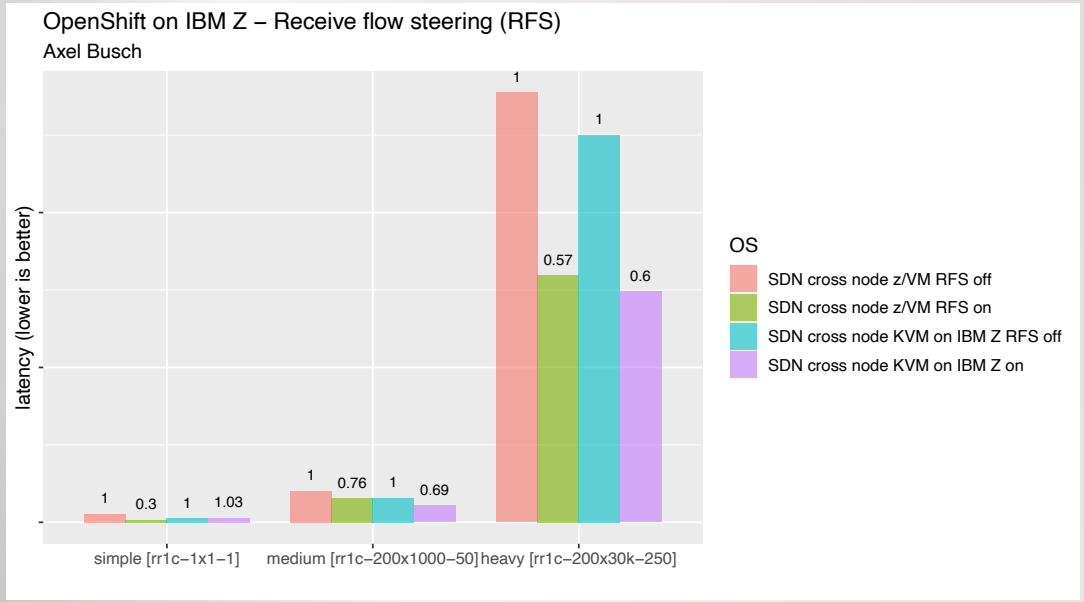


- Deployment of pods transferring data is crucial for the resulting service performance
- **Latency can go *up* to 5x-6x** comparing same node and cross node scenarios
- **Throughput can go *down* to 3%** comparing same node and cross node scenarios

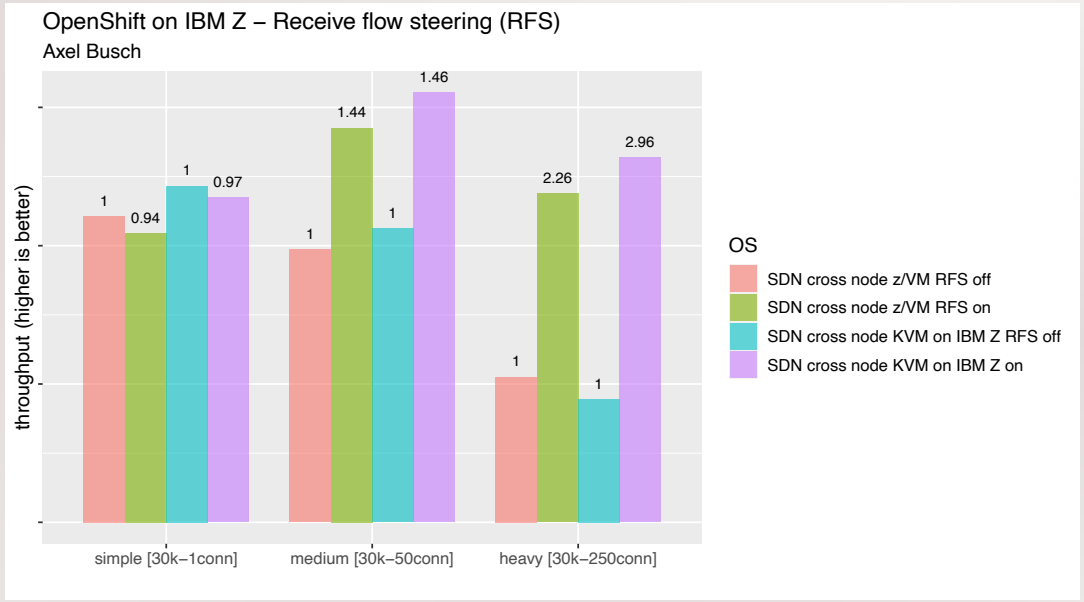
Network tunings: Receive flow steering on/off

- Especially beneficial in **non co-located scenarios**
  - Latency up to factor 3.33x better
  - Throughput up to factor 2.96x better
- Benefit especially for **medium and large** number of parallel **connections**
- For both, z/VM and KVM on IBM Z

latency



throughput



CNI Plugins: Default-SDN vs. IPVLAN vs. Host-device

- **Cross node** deployment
- With the exception of streaming workloads, both **IPVLan L2 and host-device are always better than Default-SDN**

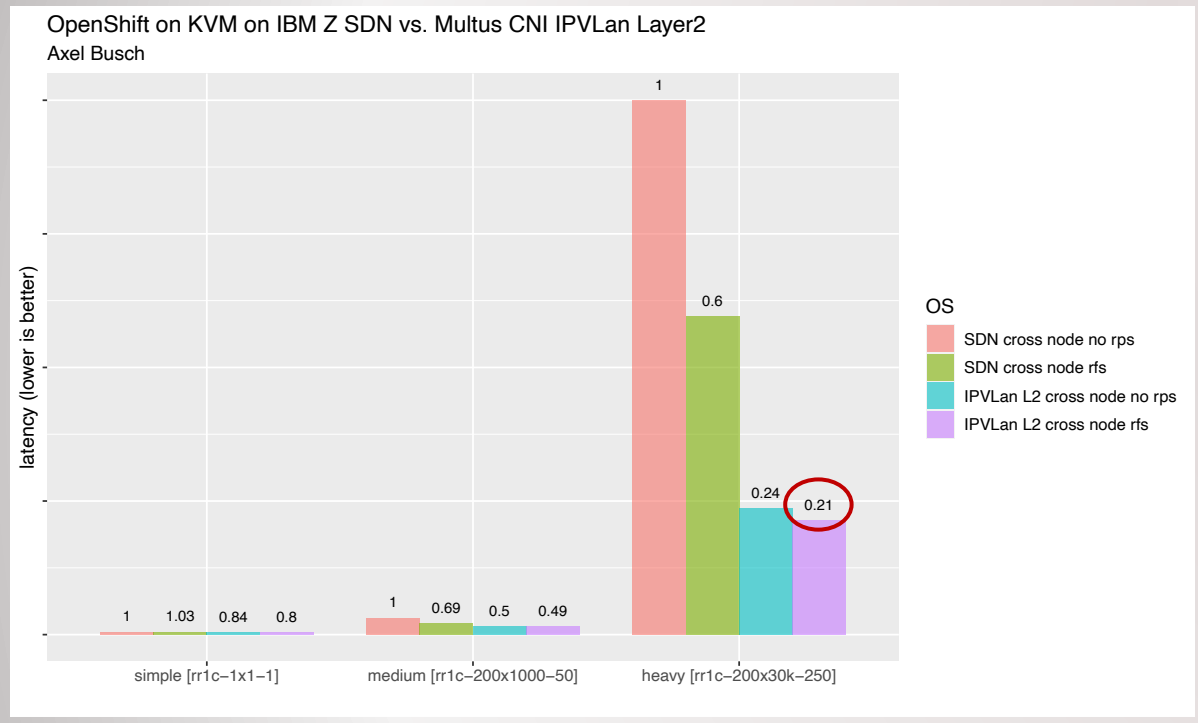
- IPVLAN vs. SDN performance:  11/15 cases

- host-device vs. SDN performance:  11/15 cases

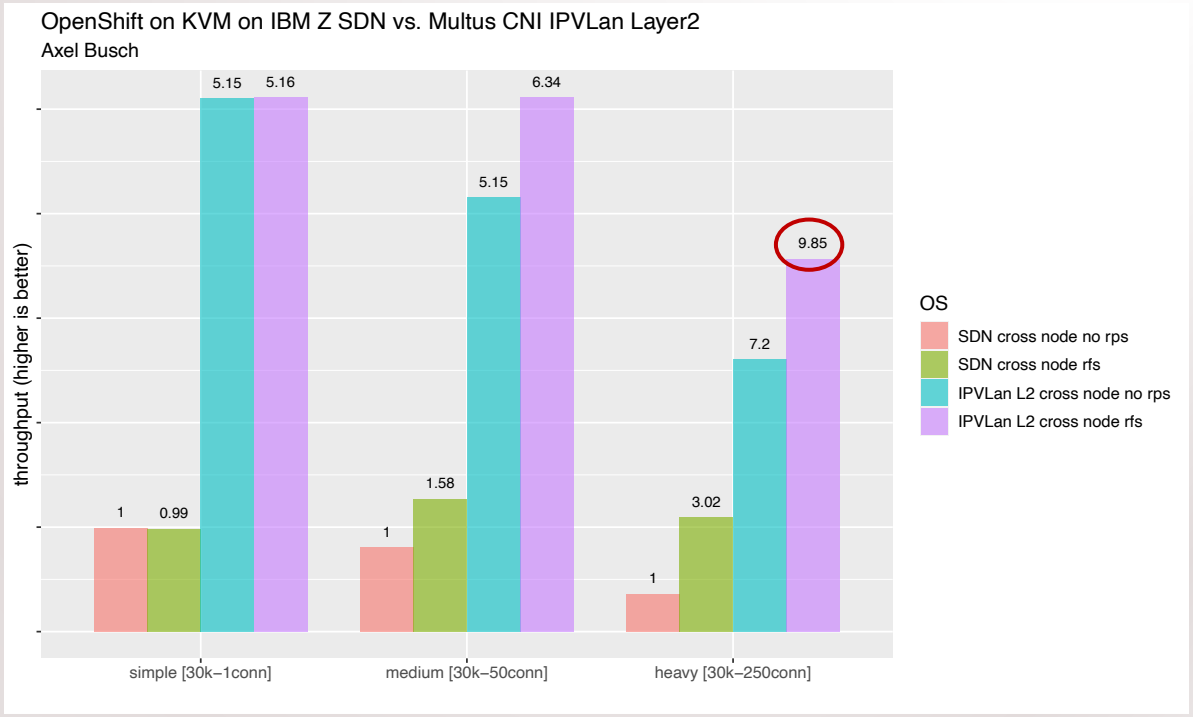
- IPVLan vs. host device?
- Usually prefer IPVLan L2 except for some scenarios.
  - **Host device** can show great performance in cases with **many connections and large packet sizes**
  - **Host device can be much more efficient compared to Default-SDN as well as IPVLan L2**
  - On KVM on IBM Z **host device shows up to 8** times more efficiency **compared to Default-SDN**
  - On KVM on IBM Z **host device can show up to 3** times more efficiency **compared to IPVLan L2**

Let's see some values: Default-SDN vs. IPVLan L2 – latency & throughput

latency



throughput



- IPVLan L2 especially beneficial in **non-colocated scenarios**
- Useful for selected pods with **high networking performance requirements**

- **Latency** improvement up to **factor 5x** (relative to SDN performance)
- **Throughput** improvement up to factor **9.85x** (relative to SDN performance)

Guideline: How to decide what to choose?

- Some ideas for **default values**:
- Low maintainance overhead
- Low-moderate networking load
- No high latency critical services
- Small chain of services

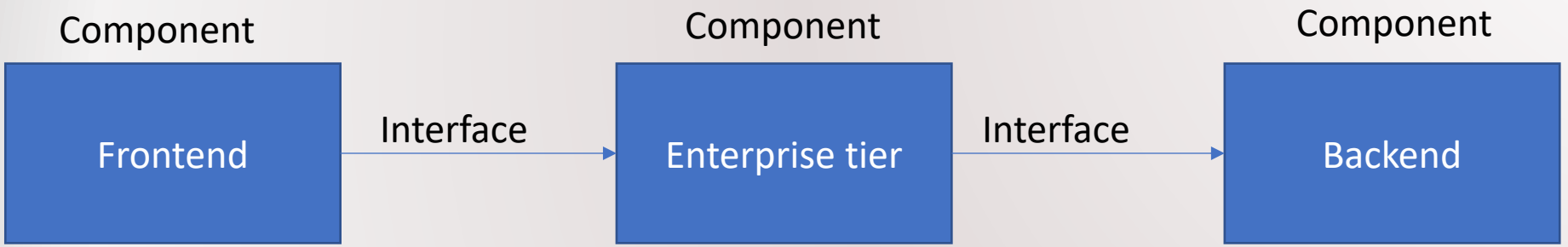
Use OpenShift-SDN (default), turn RFS on and use 2-3 infra nodes

- For other requirements such as long service chains or high networking load use scenario-based tuning ideas
- **Focus on technologies** may be **too imprecise** for own application purpose.
- Nevertheless, important to understand impact on performance and efficiency
- **Used as a basis to make right decisions** regarding project **requirements**

Consider scenario-based tuning ideas in addition

Scenario-based networking performance tuning ideas

Architecture-Style: Service-oriented Architecture

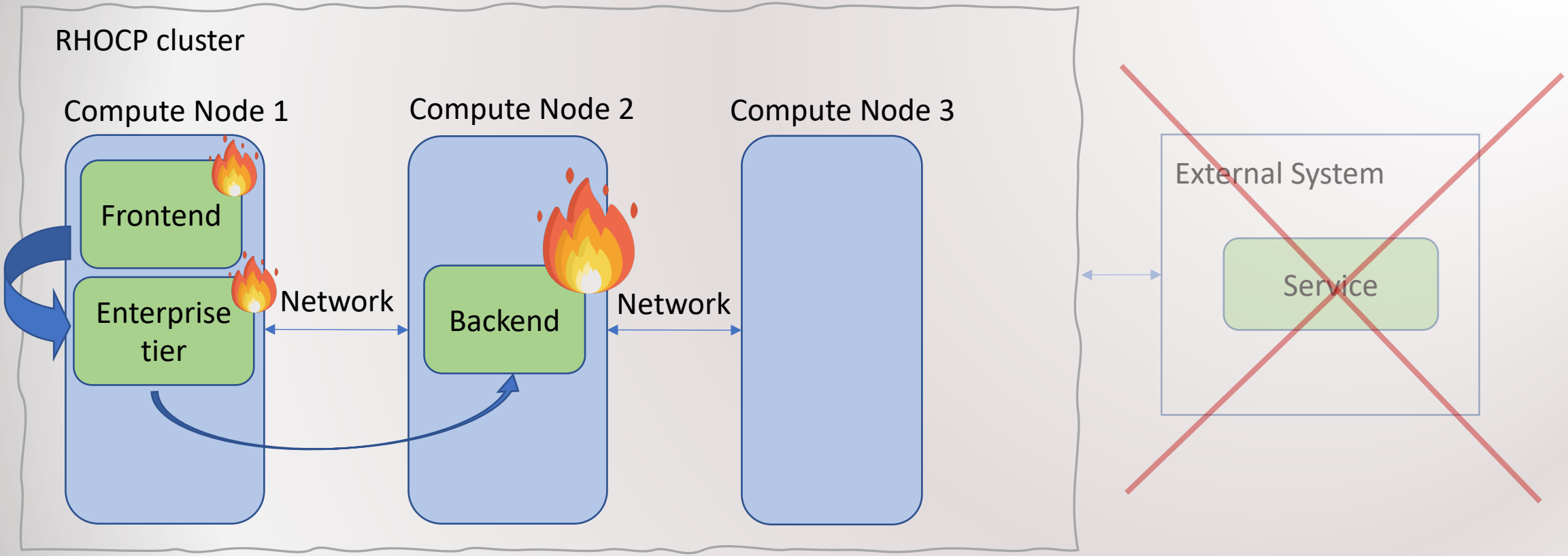


- WebGUI
- Business API
- API (for m2m queries)

- User Authentication
- Business processes
- Business logic
- Presentation logic
- ...

- Database
- Data access objects
- Data Persistency
- ...

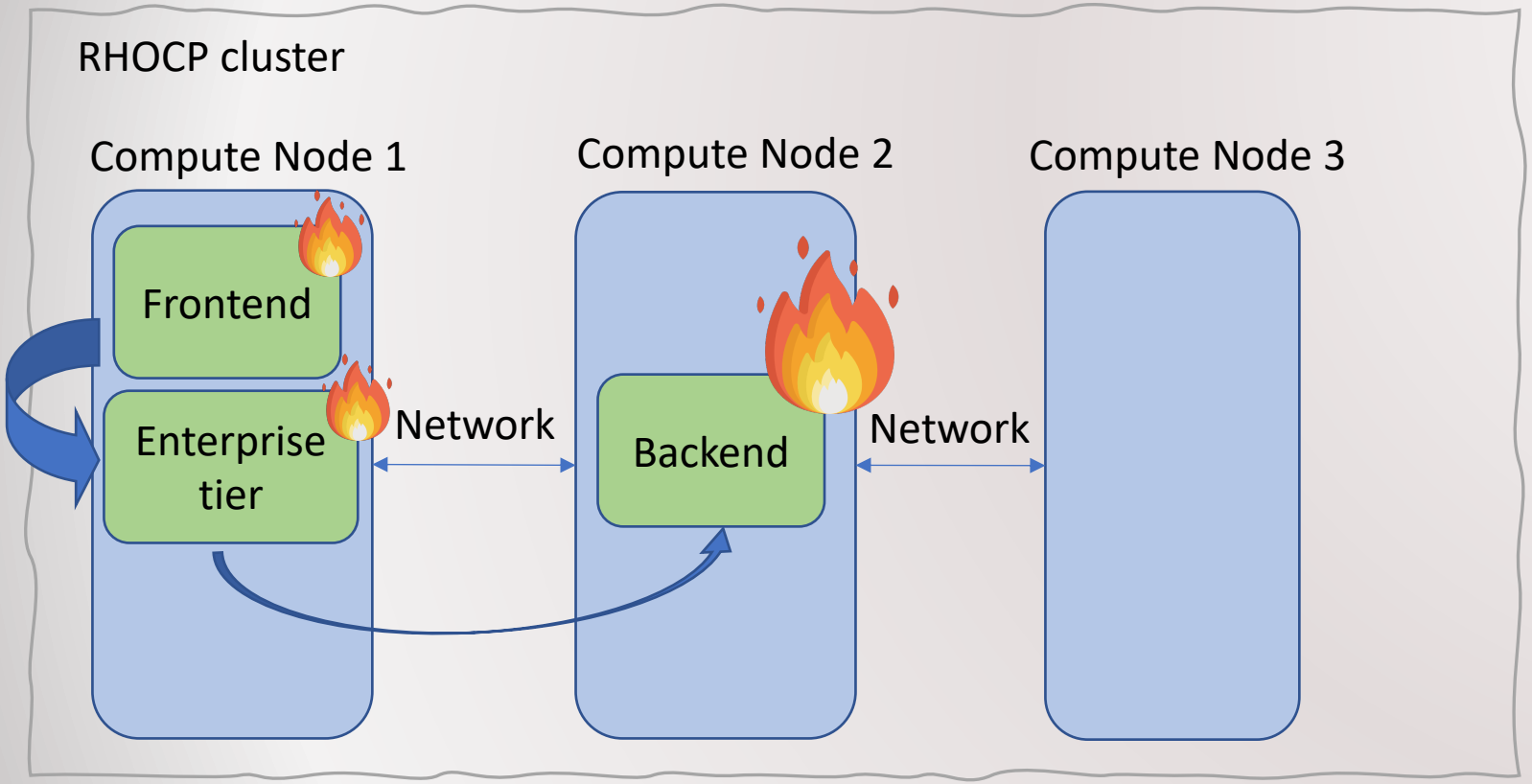
Scenario I: Low database intensive & no external (legacy) systems involved



Assumptions:

- Frontend and Enterprise tier **communicate frequently**
- Enterprise tier and Backend **do not communicate frequently**
- Backend has quite high CPU/memory requirements -> needs to be on worker with enough free resources
- **No external service required**

Scenario I: Focus on frontend & enterprise pod performance and backend CPU demands



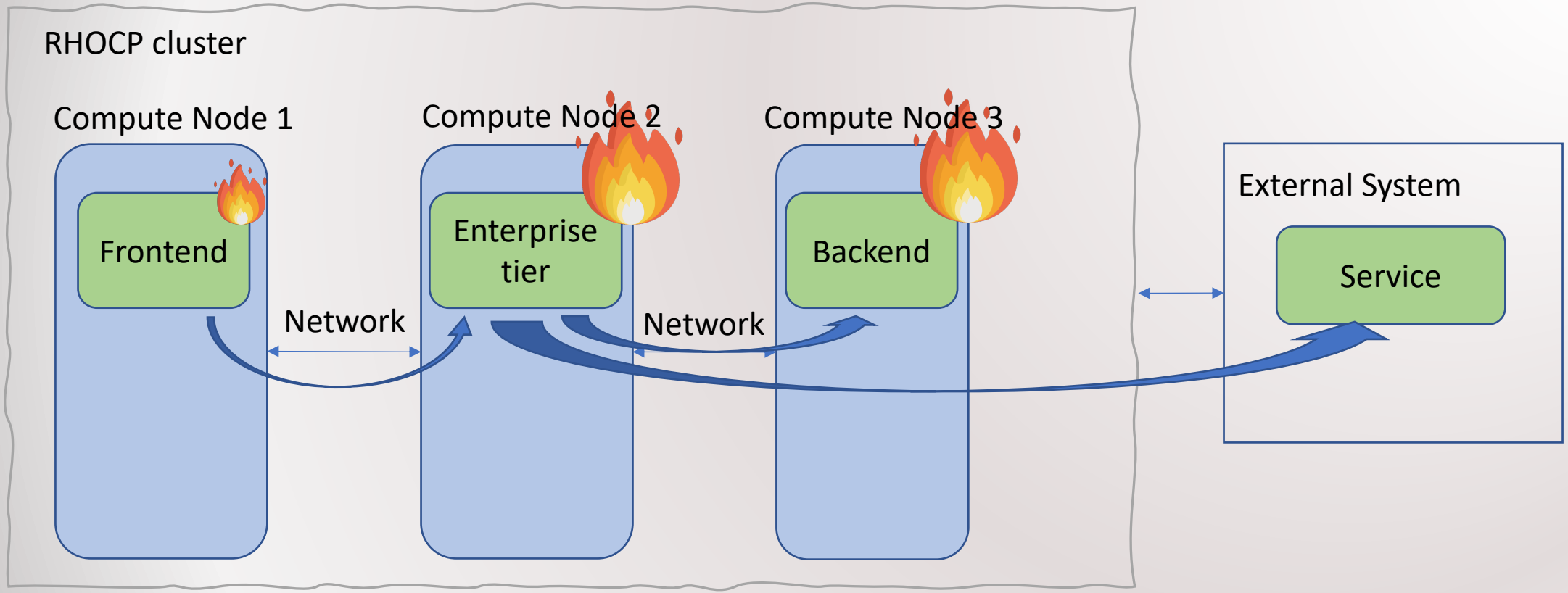
For frontend to enterprise tier:

- Use **SDN** for their communication as **fastest and most efficient** way
- z/VM more efficient and faster than KVM

For enterprise and backend:

- Use **IPVLan Layer2 especially to get best latency**
- With degradation in throughput and latency SDN can be used
- For higher efficiency, better throughput and latency **enable RFS**

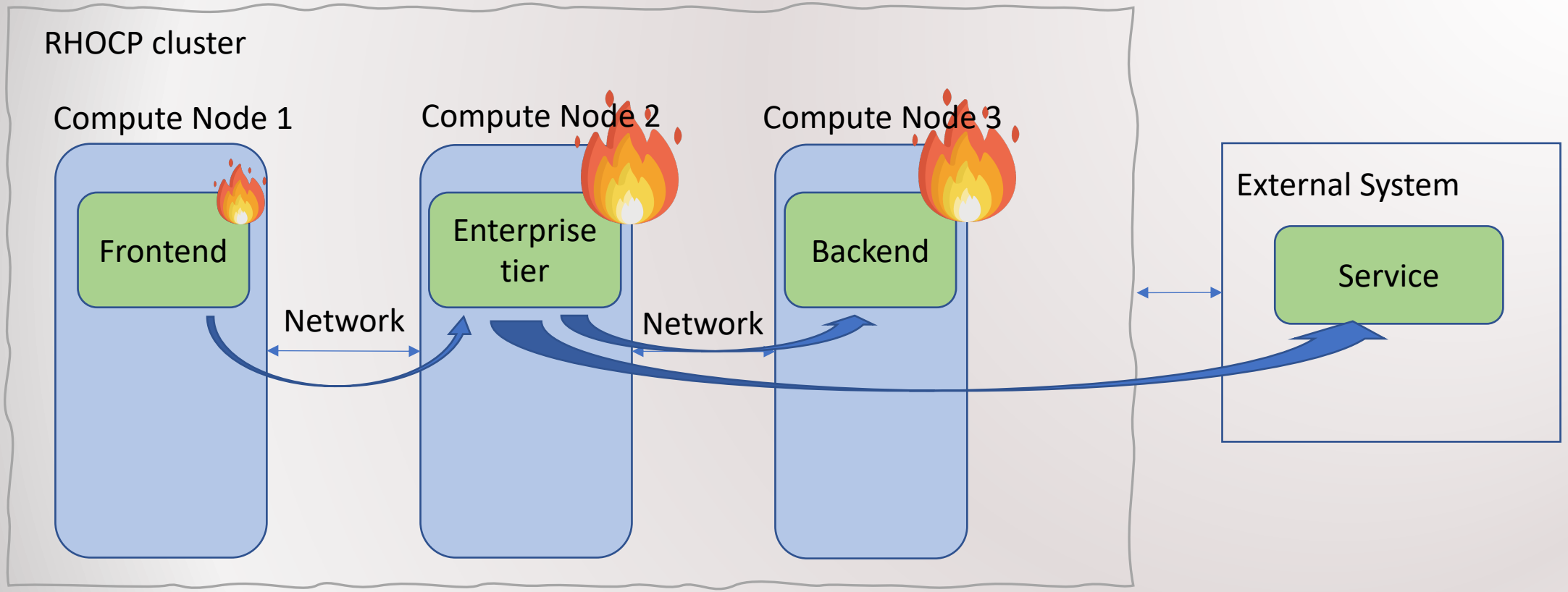
Scenario II: Highly database intensive & external (legacy) systems involved



Assumptions:

- Frontend and Enterprise tier do **not communicate** in a significant manor
- Enterprise tier and Backend **do communicate frequently (high throughput)**
- **Backend and external service communicate latency as well as throughput critical**
- Backend and Enterprise have quite high CPU/memory demands

Scenario II: Highly database intensive & external (legacy) systems involved

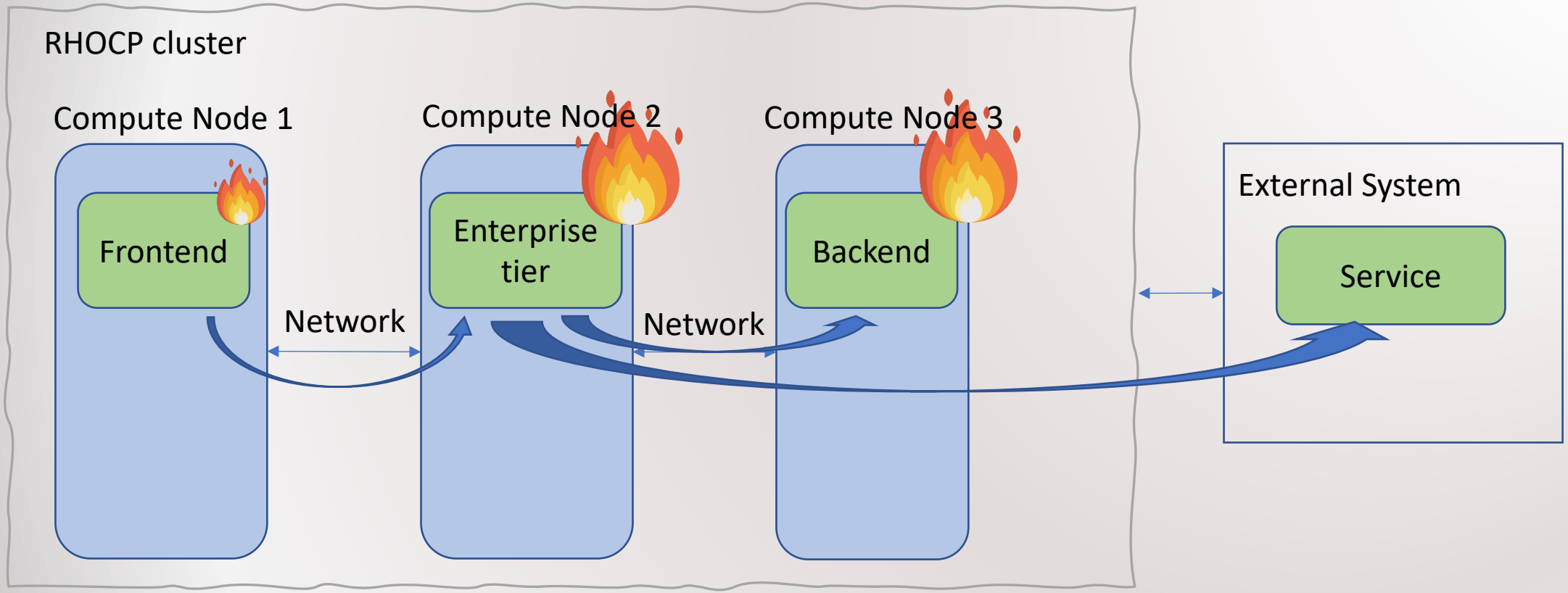


For frontend to enterprise tier:

- Use **SDN** if frontend uses just a few connections in parallel
- Use Multus **IPVLan Layer2 for many parallel connections** as second interface
- Use **RFS** in case of parallel connections (keep an eye on your CPU capacity)

- If **throughput is critical**
  - Deploy frontend and enterprise tier on same node (factor 3x-4x higher throughput possible)
  - Might require to add vCPUs to node 2

Scenario II: Highly database intensive & external (legacy) systems involved



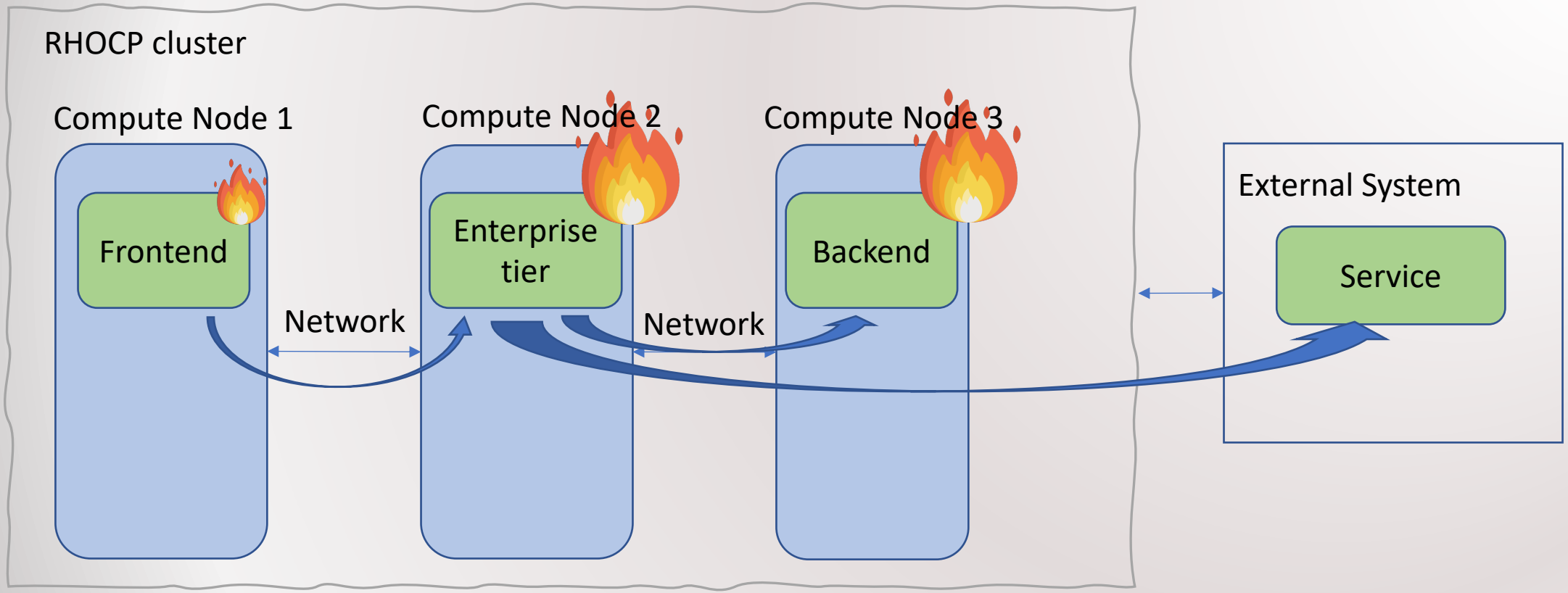
For enterprise tier to backend:

- Use Multus **IPVLan Layer2** as second interface
- Use **IPVLan Layer2 for many parallel connections**
- Use **RFS** (keep an eye on your CPU capacity)

For enterprise tier to external service:

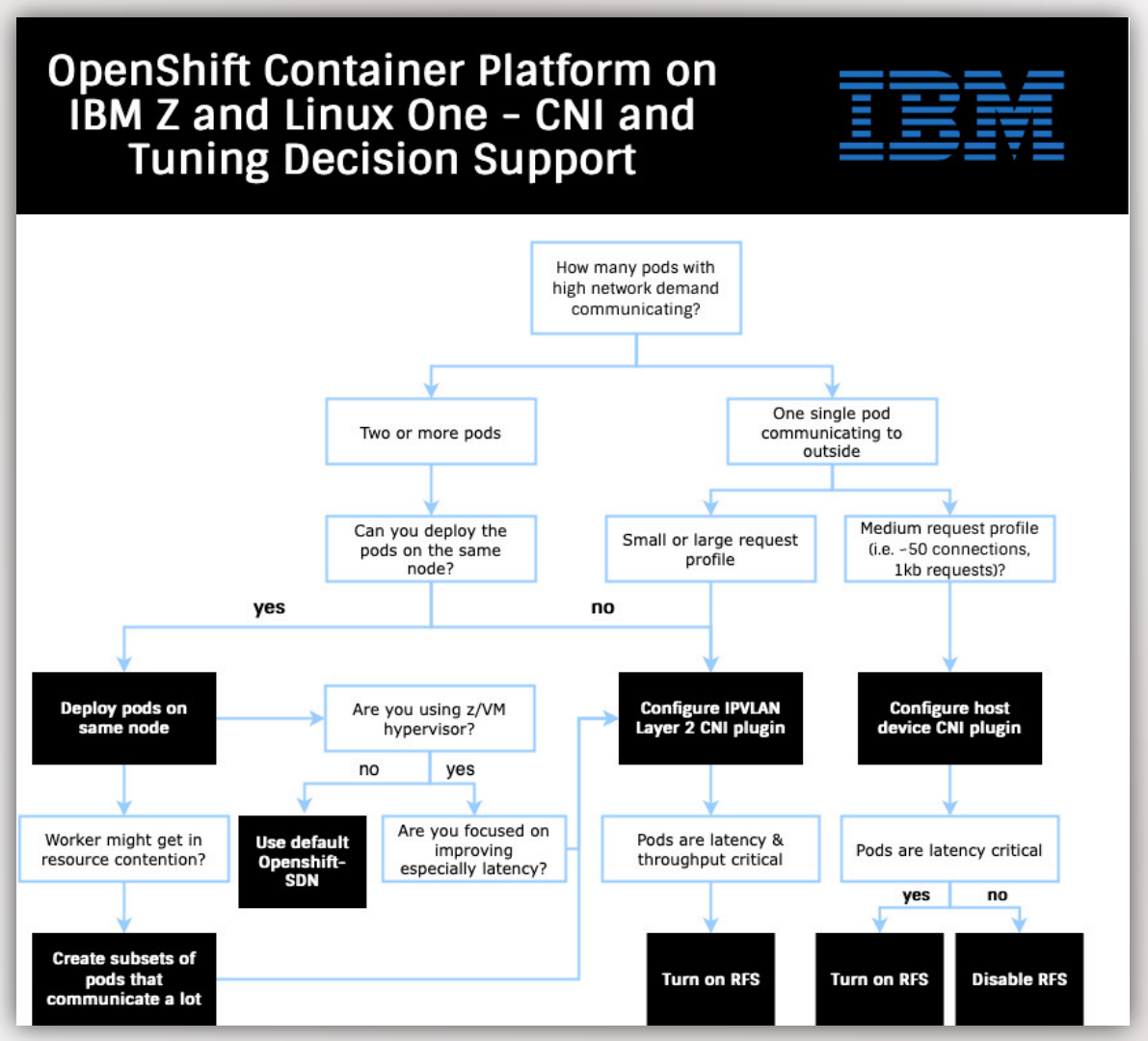
- Use Multus **host device** as third interface
- Use **IPVLan Layer2 for many parallel connections**
- Use **RFS** for many connections (keep an eye on CPU contentions)
- For only 50 connections switch RFS off

Scenario II: Highly database intensive & external (legacy) systems involved



- **Combination** of SDN, Multus IPVLan Layer2 and host device plugin **use different resources of the cluster as best as possible**.
- Highest possible throughput and lowest possible latency for particular pod requirements

Rules of thumb: When which technology/settings?



Summary & Takeaways

Summary

- Introduced the trade-off decisions to be done
- Outlined RHOCP networking architecture and how it works together with Multus CNI, how the plugings work and how they can be used
- General performance tuning ideas: When to use which CNI plugin and tunings
- Scenario-based tuning ideas for RHOCP tuning according to the individual workload and cloud environment

Take aways

- Multus CNI can be used to improve networking performance of RHOCP clusters
- Plugins show different performance attributes for different scenarios, e.g. deployment, workload, system setup
- Combination of plugins and tunings such as RFS, infra nodes must be defined according to the configuration of individual workload and cloud environment

Thank you!

Dr.-Ing. Axel Busch
Chapter Lead OCP Performance on IBM Z
axel.busch@ibm.com
Linux on IBM Z Performance