# Linux on IBM zSystems with RoCE Express

Stefan Raspl

Linux & Virtualization on IBM zSystems

# Transition to PCIe-based adapters like RoCE Express as the strategic adapter for Linux on zSystems

**Statement of Direction**

*"In the future, IBM plans to shift from* ***OSA-Express*** *to* ***PCIe-based networking devices like RoCE Express*** *as the target strategic adapter type for IBM Z direct access networking connection to Linux operating systems.*

*[...] Linux on IBM Z clients that indirectly access the OSA-Express adapter family through the* ***z/VM Virtual Switch (VSwitch)*** *will be* ***unaffected*** *by this change.*

*Linux on IBM Z networking currently supports two Ethernet networking connectivity options: the OSA-Express adapter family and the RoCE Express adapter family. Use of PCIe-based networking devices as provided by the RoCE Express adapter family is aligned with the deployment model for Linux on other architectural platforms, facilitates use of broader existing Linux ecosystem tooling, and eases the effort to enable exploitation of industry hardware optimizations and integrate into industry software-defined networking models and tools, including Red Hat OpenShift Container Platform (OCP).*

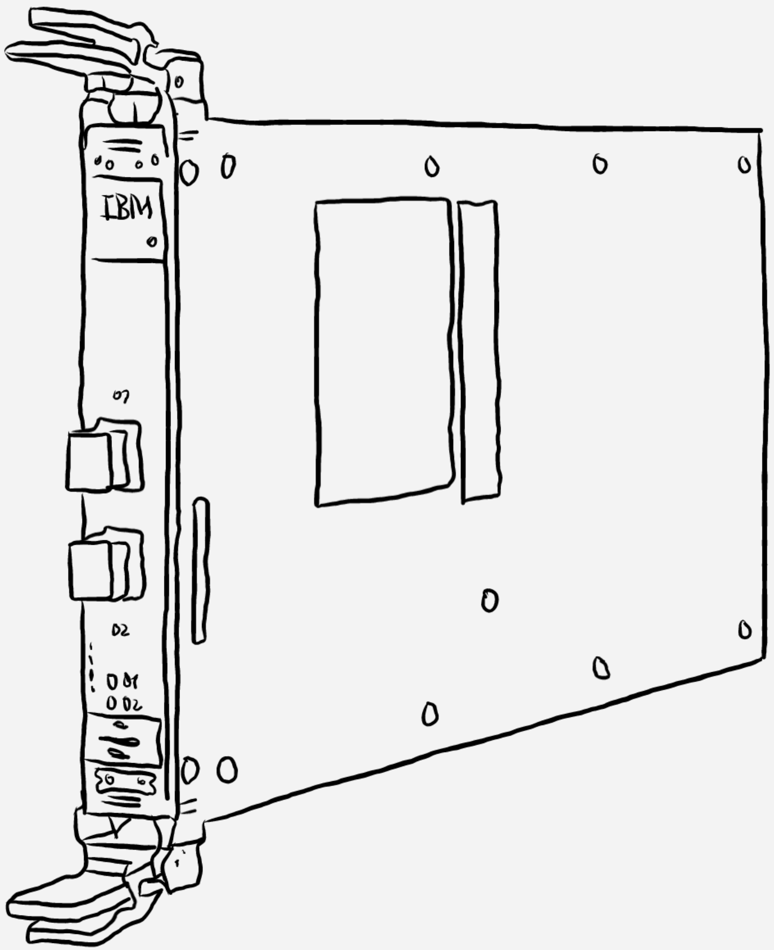
***Clients are strongly encouraged to plan accordingly for their adoption of RoCE Express adapters for IBM Z networking connectivity.***

*IBM plans to continue to work toward* ***common networking adapters for all operating systems*** *on IBM Z, IBM LinuxONE, and Linux on IBM Z."*

***Source****: IBM z16 Announcement Letter*

# Agenda

- The Cards
  - Models & Features
  - Virtualization Capabilities
- **Device Drivers, Features and Commands**
- **Performance**
- **Summary**
- **References**



# Overview

- **Introduced with zEC12 for SMC-R**
- **10 and 25GbE models, optical connectors only**
- **25GbE model strictly requires 25GbE capable switch – no negotiation to 10GbE**
- **All models feature 2 ports**
- **Fiber optics only**
- **TCP/IP[1] or RoCE (RDMA over Converged Ethernet)**
- **TCP/IP functionality exploited by Linux only**

| Feature | z16 | z15 | z14 |
|---|---|---|---|
| ***RoCE Express 3*** | 25 GbE<br>10 GbE | | |
| ***RoCE Express 2.1*** | 25 GbE<br>10 GbE | 25 GbE<br>10 GbE | |
| ***RoCE Express 2*** | 25 GbE<br>10 GbE | 25 GbE<br>10 GbE | 25 GbE<br>10 GbE |
| ***RoCE Express*** | | 10 GbE | 10 GbE |

**[1]** *Synonymous to any kind of "traditional" network traffic (TCP, UDP, etc.)*

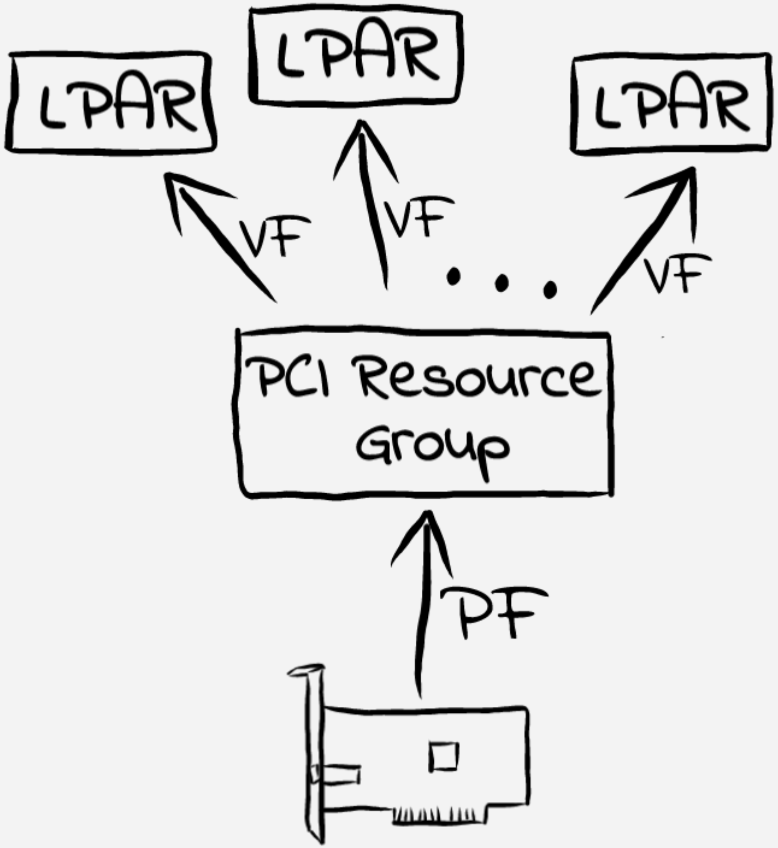
# Hardware Features

- Regular **PCI device**
  **⇒ Few IBM Z-specific tooling required**
- **Two modes of operation**:
  - TCP/IP
  - RDMA over Converged Ethernet (RoCE)
- Mode chosen by software – can be used in parallel

- **Features** (selection)
  - HW offloads: Checksumming, TSO
  - RDMA over Converged Ethernet (RoCE)
  - Flow Control, Explicit Congestion Notification
  - IPoIB, uDAPL, et al
  - VLAN, QoS, et al
- **RAS**
  - Regular RAS capabilities
  - Changing optics of a single card disrupts entire PCHID
  - Firmware updates are disruptive

# Virtualization

- Single Root I/O Virtualization (***PCI SR-IOV***)
- **Virtual Functions** (***VFs***) provide a limited subset of card functionality
- **Physical function** (***PF***) held by one of four PCI Resource Groups
  ⇒ *required for certain functionalities, including firmware updates and promiscuous mode enablement of VFs*
- ***RoCE Express***
  - Up to 31 VFs per card(!)
  - Each VF identifies the entire card / both ports
- ***RoCE Express2***
  - Up to 63 VFs per port
  - Each VF identifies a single port

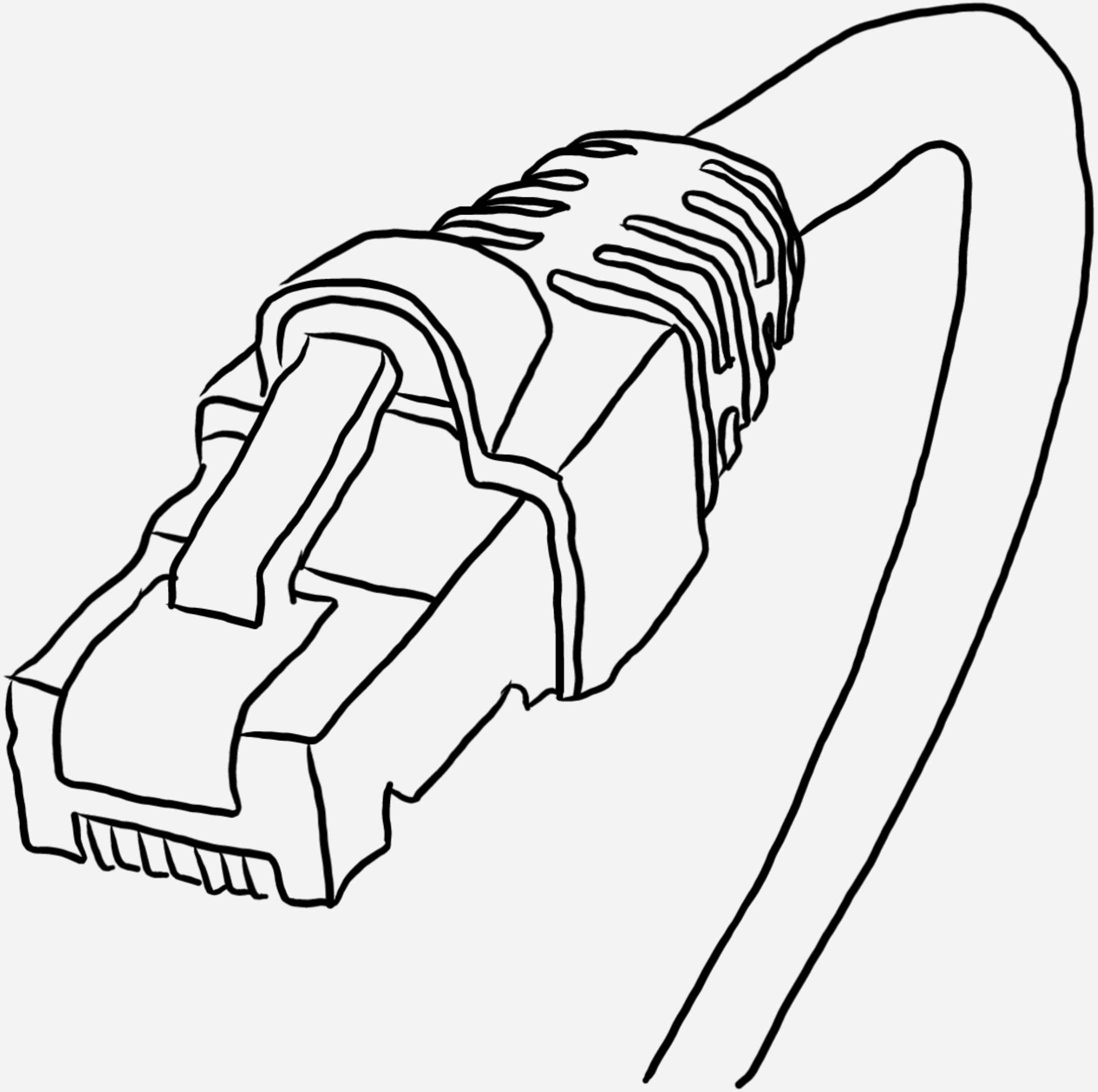


# Virtualization Capabilities

| **Model** | ***#Cards*** | ***#Ports / Card*** | ***Total #Ports*** | ***#IP Stacks / Card*** |
|---|---|---|---|---|
| **z16** | 16 | 2 | 32 | 31-126 |
| **z15** | 16 | 2 | 32 | 31-126 |
| **z14** | 8 | 2 | 16 | 31-126 |
| **z14 ZR1** | 4 | 2 | 8 | 31-126 |

# Agenda

- **The Cards**
- **Device Drivers, Features and Commands**
  - Installation
  - Distro Support
  - Tools
  - Device Drivers
- **Performance**
- **Summary**
- **References**



# Identifying PCI Devices

- **Function Identifier (FID)**
  - Mandatory for every VF/device
  - Unique per CPC
  - Not portable - makes migration either within CPC or across CPCs complicated!
  - Required for device activation
  - **Note**: For RoCE Express, the FID identifies the ***card***, while for RoCE Express2 the FID identifies ***port***.
- **User Identifier (UID)**
  - Optional – if not set in IOCDS, UIDs are assigned automatically, but not persistent!
  - Unique within partition only
- **User Identifier (UID) - *continued***
  - Well suited for migrations
  - Assign UIDs based on conventions, e.g. always use UID 1 in each partition for main networking interface
  - Reflected as Domain in PCI ID: `0001:00:00.0`

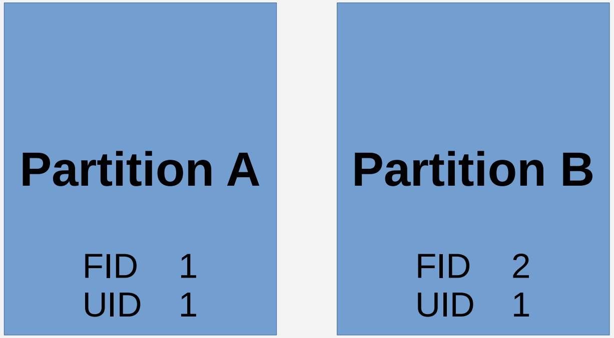



**Fig.1**: *FIDs are unique within a CPC, UIDs are unique within a partition*

# PCI Device UIDs

- **Assigning unique UIDs is *strictly required* for Linux on Z!**
  - Otherwise, interface names might change between reboots, and are hard to predict
- **Enable uniqueness enforcing**
  - Guarantees availability and uniqueness of UIDs for all PCI devices
  - Check state via `sysfs`[1]:

```
root:~# cat /sys/bus/pci/devices/<ID>/uid_is_unique
1
```

  - **z/VM** (default: **on**)

```
#CP SET IO_OPT UID ON
```

**Beware**: Can be enabled *dynamically*, but that is not recommended

- **KVM**

UIDs mandatory for all PCI device definitions in domain XML

```
<address type='pci' domain='0x0002' bus='0x00'
         slot='0x01' function='0x0'>
   <zpci uid='0x0001' fid='0x00000000'/>
</address>
```

- **LPAR**
  - DPM: Always enabled
  - Classic Mode: Toggle via HCD (default: **off**):

```
                        Add Partition

  Specify the following values.

  Partition name . . . LP01____
  Partition number . . 1      (same as MIF image ID)
  Partition usage  . . OS     +
  UID uniqueness . . . Y      (Y/N)

  Description  . . . . Test partition 01____________
```

[1] *Linux Kernel 5.13 or later*

# PCI Device Activation

- **Use *Candidate List* to define partitions that can configure a FID online at all**
- **Every device needs to be online to become visible in a partition:**

```
$ echo 1 > /sys/bus/pci/slots/<FID>/power
```

- **First come, first serve: First partition to set a device online gains exclusive access!**
- **Configuration state persistent across IMLs**
- **However: PCI FIDs required for fresh install need to be online *prior* to installation!**

- **Passthrough devices in z/VM and KVM guests are always auto-activated**
- **Activation for new installs in LPAR:**
  - **Alternative 1** (Classic mode only)
    Use ***Access List*** to set a device online for a specific partition on partition activation.
  - **Alternative 2** (Classic or DPM mode)
    Set device online on Support Element (see next page)
  - **Alternative 3** (DPM mode only)
    - Auto-activation in Linux during boot
    - Distro integration: RHEL 8.6, Ubuntu 22.04

# PCI Device Activation on *Support Element*

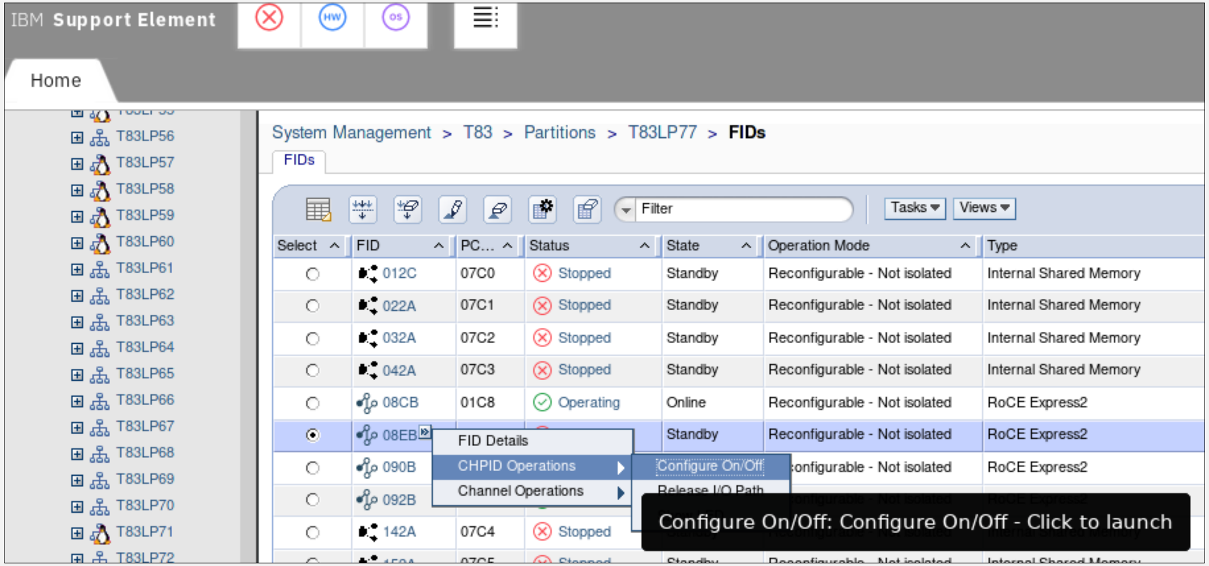



**Fig.1**: *Select CHPID Operations > Configure On/Off*

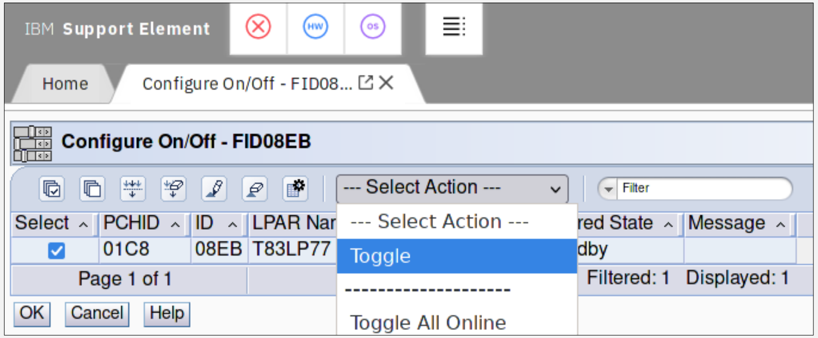



**Fig.2**: *Toggle online state for the device*

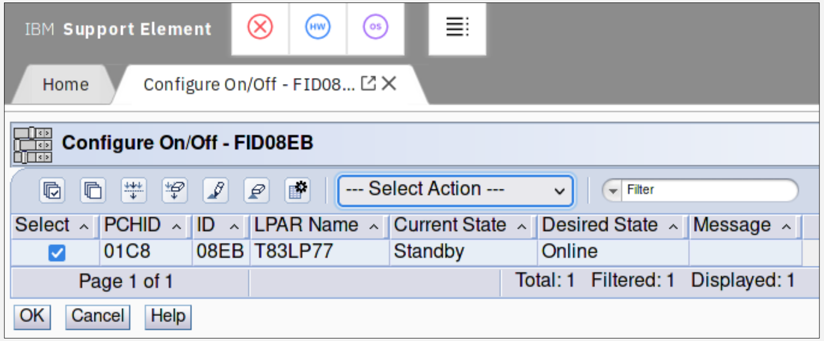



**Fig.3**: *Verify state transition from “Standby” to “Online” prior to execution*

# Predictable Interface Names

- **A priori knowledge of interface name required for RoCE-only LPAR installs with some Linux distributions**
- **`Systemd` assigns interface names based on PCI properties according to a hierarchy of rules**
- **New, *truly* predictable naming schemes starting with** *Linux kernel 5.13* **and** *Systemd v249***:**
  - With unique UIDs enabled (*recommended*):

    ```
    eno<UID_in_dec>
    ```

  - With unique UIDs disabled:

    ```
    ens<FID_in_dec>
    ```

  **Beware: This is UID/FID in *decimal*!!**
- **Distro Support**: RHEL 9, Ubuntu 22.04
- **Hint**: Use UID range `0x1` - `0x9` for your interfaces to avoid the need to convert between decimal and hex values

```
ro ramdisk_size=60000 cio_ignore=all,!condev,!1900,!1940
ip=192.168.91.48::192.168.91.49:16:<lpar>:vlan210:none
nameserver=192.168.91.49 domain=<domain>
vlan=vlan210:enP6p0s0d1 inst.repo=... inst.ssh=1
sshpassword=xxx inst.vnc=1 inst.vncpassword=xxx
```

**Fig.1**: *Interface name specified as part of PRM file on RHEL*

**Previously**, interface names could be any of:

- `ens<FID>`
- `enP<UID>s<FID>`
- `enP<UID>p0s`
- `enP<UID>p0np0`
- `enP<UID>p0s0np0`
- `enP<UID>p0`

Selection criteria would depend upon, among others: UID value, FID being all numeric or not, kernel level, et al.

**Fig.2**: *Interface naming prior to Systemd v249*

# Linux Distribution Installation

1. **Make sure PCI FID is activated for installation**
2. **Specify PCI device in PRM file**
   **Note**: RoCE Express adapters are PCI devices – mechanisms to configure network in PRM file from OSA-Express do not apply!
   - **RHEL**: Reference PCI FID via interface name in `ip` statement

     ```
     ip=10.10.92.31::10.10.92.30:16:<lpar>:enP540p0s0:none
     ```

     (use predictable interface names with RHEL 9 or later)
   - **SLES**: Reference PCI FID using interface name `ethX` as usual
   - **Ubuntu**: See RHEL (use predictable interface names with Ubuntu 22.04 or later)
3. **Remaining installation steps as usual**

# Installers: Interface Name Workarounds

- **Problem: How to tell interface name if predictable interface names are *not* available?**
- **Approach:**
  - Pick a random interface name
  - Observe the installation messages on the HMC for the correct name, and start over

```
[...]
[22.04198] mlx5_core 000a:00:00.0: enabling device (0000 -> 0002)
[22.04203] mlx5_core 000a:00:00.0: firmware version: 14.25.1020
[22.26202] mlx5_core 000b:00:00.0: enabling device (0000 -> 0002)
[22.26211] mlx5_core 000b:00:00.0: firmware version: 14.25.1020
[22.48356] mlx5_core 0008:00:00.0: MLX5E: StrdRq(0) RqSz(1024) [...]
[22.63801] mlx5_core 0009:00:00.0: MLX5E: StrdRq(0) RqSz(1024) [...]
[22.82587] mlx5_core 000a:00:00.0: MLX5E: StrdRq(0) RqSz(1024) [...]
[22.98716] mlx5_core 000b:00:00.0: MLX5E: StrdRq(0) RqSz(1024) [...]
[23.35750] mlx5_core 0008:00:00.0: enP8p0s0: renamed from eth0
```

- **Alternative: Disable predictable interface names via kernel command line:**

  ```
  net.ifnames=0
  ```

  - Interfaces will adopt `ethX` naming scheme
  - Only practical in case of a single networking interface (or two in case of bonding)
  - Not recommended, as this will cause problems as soon as further interfaces are added

# Required Software Levels

- **RoCE Express**
  - Device Driver: mlx4
  - Linux distribution support:
    - RHEL 7 or later
    - SLES 12 or later
    - Ubuntu 16.04 LTS or later
- **RoCE Express2**
  - Device Driver: mlx5
  - Linux distribution support:
    - RHEL 7.3 with service or later
    - SLES 12 SP3 with service or later
    - Ubuntu 16.04 LTS with service or later
- **RoCE Express3**
  - Device Driver: mlx5
  - Linux distribution support:
    - RHEL 7.9, 8.4 with service or later
    - SLES 12 SP5, 15 SP3 with service or later
    - Ubuntu 20.04, 22.04 LTS with service or later
- **z/VM**
  - v6.3 with service or later for PCI passthrough support
- **KVM**
  - Passthrough support in progress

# Tooling: smc_rnics

- **Quick PCI networking devices overview**
- **Use option –I to switch to InfiniBand-related display**
- **Note**:
  - Ethernet ports start at 0, while RDMA device ports start at 1!
  - RDMA considers each port as separate device – hence all ports are 1!
- **Illustrates relationship of** FID ⇔ PCI ID ⇔ PCHID ⇔ interface name ⇔ IB device name
- **Also supports device activation/ deactivation**

```
root:~# smc_rnics
  FID Power PCI_ID       PCHID Type          PPrt PNET_ID Net-Dev
-------------------------------------------------------------------
  8ca 1     0008:00:00.0 01c8  RoCE_Express2 0    NET25   enP8p0s0
  8ea 0
  90a 1     000a:00:00.0 01cc  RoCE_Express2 0    NET25   enP10p0s0
  92a 1     000b:00:00.0 01cc  RoCE_Express2 1    NET26   enP11p0s0

root:~# smc_rnics -e 8ea

root:~# smc_rnics
  FID Power PCI_ID       PCHID Type          PPrt PNET_ID Net-Dev
-------------------------------------------------------------------
  8ca 1     0008:00:00.0 01c8  RoCE_Express2 0    NET25   enP8p0s0
  8ea 1     0009:00:00.0 01c8  RoCE_Express2 1    NET26   enP9p0s0
  90a 1     000a:00:00.0 01cc  RoCE_Express2 0    NET25   enP10p0s0
  92a 1     000b:00:00.0 01cc  RoCE_Express2 1    NET26   enP11p0s0

root:~# smc_rnics -I
  FID Power PCI_ID       PCHID Type          IPrt PNET_ID IB-Dev
--------------------------------------------------------------------
  8ca 1     0008:00:00.0 01c8  RoCE_Express2 1    NET25   mlx5_0
  8ea 1     0009:00:00.0 01c8  RoCE_Express2 1    NET26   mlx5_1
  90a 1     000a:00:00.0 01cc  RoCE_Express2 1    NET25   mlx5_2
  92a 1     000b:00:00.0 01cc  RoCE_Express2 1    NET26   mlx5_3
```

**Fig.1**: *Sample* smc_rnics *output*

# Tooling: `ethtool`

- **Display and change Ethernet device driver and hardware settings**
- **Information provided (among others):**
  - Link speed
  - carrier status
  - Firmware version
- **Use options `–k` and `–K` to query and/or set device features like checksum offloads or TCP segmentation offload**

```
root:~# ethtool enP9p0s0
Settings for enP9p0s0:
        Supported ports: [ FIBRE Backplane ]
        Supported link modes:   1000baseKX/Full
                                10000baseKR/Full
        [...]
        Speed: 25000Mb/s
        [...]
        Link detected: yes

root@~# ethtool -i enP9p0s0
driver: mlx5_core
version: 5.14.0-rc4sr_devel+
firmware-version: 14.25.1020 (IBM0000000016)
[...]

root:~# ethtool -k enP9p0s0
Features for enP9p0s0:
rx-checksumming: on
tx-checksumming: on
        tx-checksum-ipv4: off [fixed]
        tx-checksum-ip-generic: on
        tx-checksum-ipv6: off [fixed]
        tx-checksum-fcoe-crc: off [fixed]
        tx-checksum-sctp: off [fixed]
scatter-gather: on
        tx-scatter-gather: on
        tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: on
        tx-tcp-segmentation: on
        [...]
```

**Fig.1**: *Sample* `ethtool` *output*

# Tooling: ibv_devinfo & zpcictl

- **ibv_devinfo**
  - **Display RDMA interface information**
  - **Comes with package ibverbs-utils**
  - **Useful information: MTU sizes**
  - **Notes**:
    - MTU size for RDMA has fixed values: 256, 512, 1024, 2048 or 4096 Bytes.
    - Driver automatically chooses largest possible value depending on Ethernet MTU size (use `ip link set mtu` command to modify)
- **zpcictl**
  - Part of *s390-tools*
  - Report defective PCI devices to Service Element (SE)

```
root:~# ibv_devinfo
hca_id: mlx5_1
        transport:                      InfiniBand (0)
        fw_ver:                         14.25.1020
        node_guid:                      9928:1bfe:ff9b:1082
        sys_image_guid:                 9803:9b03:001b:2898
        vendor_id:                      0x02c9
        vendor_part_id:                 4118
        hw_ver:                         0x0
        board_id:                       IBM0000000016
        phys_port_cnt:                  1
                port:   1
                        state:                  PORT_ACTIVE (4)
                        max_mtu:                4096 (5)
                        active_mtu:             1024 (3)
                        sm_lid:                 0
                        port_lid:               0
                        port_lmc:               0x00
                        link_layer:             Ethernet
```

**Fig.1**: *Sample* `ibv_devinfo` *output*

# Performance Tuning

- **Most applicable “classic” hardware offloads like rx/tx checksumming or TCP segmentation offload enabled per default**
- **Receive Packet Steering (RPS):**
  - Distribute inbound packets more evenly across specified CPUs
  - Can take advantage of hot caches
    ⇒ Meaningful with multiple CPUs only
  - *Can* provide good performance improvements, especially with many connections and small packet sizes
- **Receive Flow Steering (RFS)**
  - Similar to RPS, but add’l consideration towards location of userspace process consuming inbound traffic

**NOTE**: As some of these options can have counter-productive effects in corner cases, ***always*** have a representative(!!!) benchmark ready to verify that you are indeed improving performance!!

# Performance Tuning (*continued*)

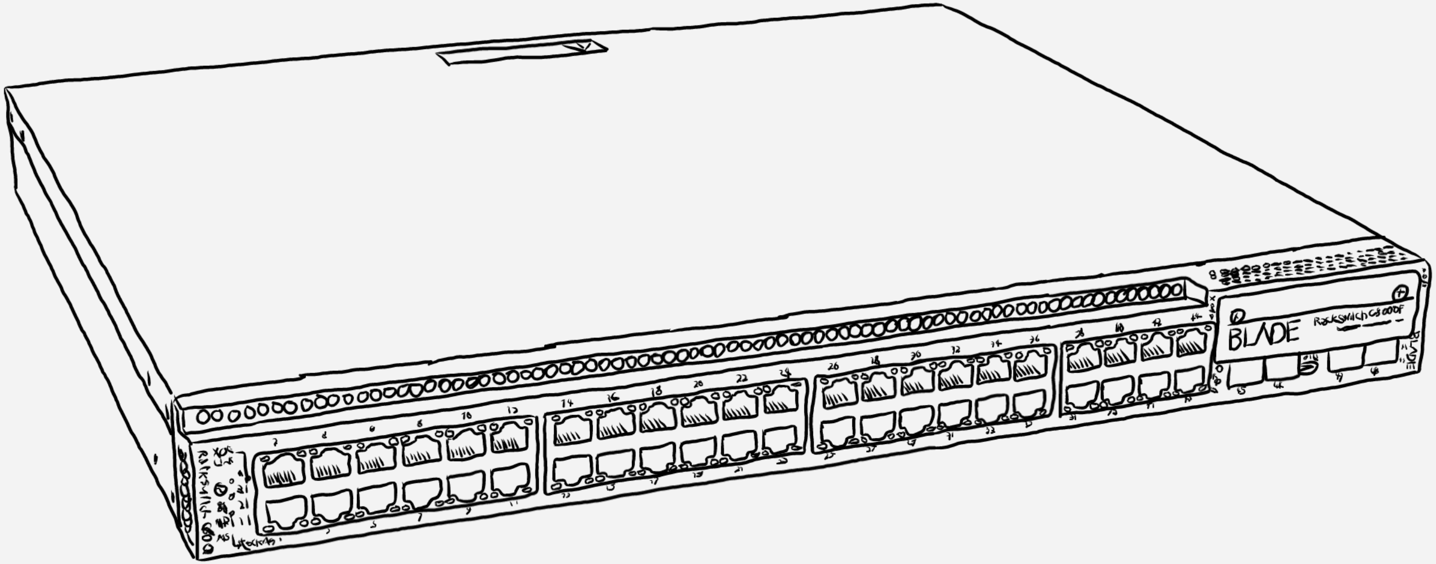
- **Enable Striding RQ**
  - More efficient use of inbound buffers
  - Helps with inbound streaming workloads.
- **Interrupt Moderation**
  - Modify waiting behavior on inbound packets before sending an interrupt
  - Use `rx/tx-usecs` to modify waiting time, and `rx/tx-frames` to modify waiting buffer count
  - Can save CPU cycles at the cost of latency and throughput, or improve latency at the cost of CPU
  - Default is usually a good compromise
  - See here for further details

```
root:~# ethtool --show-priv-flags enP9p0s0
Private flags for enP9p0s0:
rx_cqe_moder        : on
tx_cqe_moder        : off
rx_cqe_compress     : off
rx_striding_rq      : off
[...]

root:~# ethtool --set-priv-flags enP9p0s0 \
                        rx_striding_rq on

root:~# ethtool --show-priv-flags enP9p0s0
Coalesce parameters for enP10p0s0np0:
Adaptive RX: on  TX: on
stats-block-usecs: 0
sample-interval: 0
pkt-rate-low: 0
pkt-rate-high: 0

rx-usecs: 8
rx-frames: 128
[…]

root:~# ethtool -C enP10p0s0np0 rx-usecs 4095 \
          rx-frames 65535 tx-usecs 4095 tx-frames 65535
```

**Fig.1**: *Checking and setting Striding RQ enablement and interrupt moderation settings via* `ethtool`

# Agenda

- **The Cards**
- **Device Drivers, Features and Commands**
- **Deployment Considerations**
  - LPAR
  - z/VM
  - KVM
  - RDMA
- **Performance**
- **Summary**
- **References**

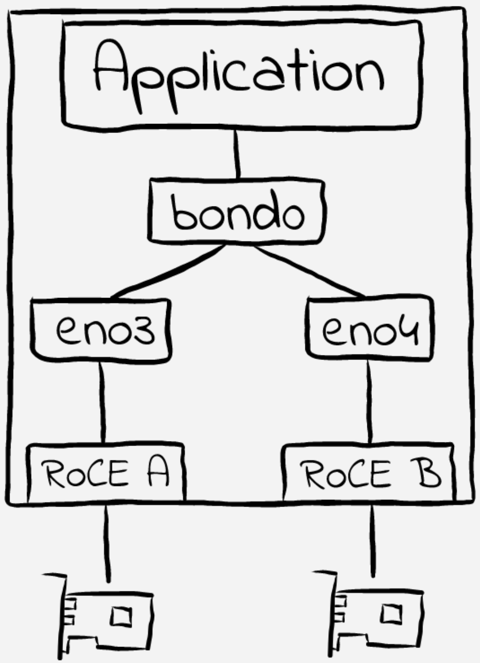


# LPAR Linux & `bonding` Driver

- LPAR configuration as usual
  - Use `ip` command for transient interface setup
  - See distribution’s manual for persistent configuration
- Use Linux **`bonding`** driver to aggregate multiple network interfaces into a single logical “bonded” interface.
- Check

```
/sys/bus/pci/devices/<ID>/pfip/segment0
```

  to verify that each PCI device belongs to a different PCI resource group!
- Various modes available, providing HA or load-balancing functionality.
- See this white paper for further details:
  Linux Channel Bonding Best Practices and Recommendations
  Note:
- Alternatives:
  - *Teaming driver*: Similar to bonding driver; **deprecated**
  - *Open vSwitch*: Provides bonding functionality (among others)




```
# Setup bonding interface with miimon option
root:~# ip link add bond0 type bond mode active-backup miimon 100 fail_over_mac active

# Connect physical devices to bonding interface
root:~# ip link set dev eth0 master bond0
root:~# ip link set dev eth1 master bond0

# Assign IP address and activate bonding interface
root:~# ip addr add 10.100.80.36/16 dev bond0
root:~# ip link set dev bond0 up
```

**Fig.1**: *Sample bonding setup (transient)*

# Shared Port Traffic

- **Assign VFs defined on same physical port to co-located Linux images for in-adapter switching**
- **Excellent throughput**
- **Effectively provides converged interface:**
  - Provides external connectivity
  - Take advantage of co-location where possible
- **Note**: Shared IP traffic works with Linux images only due to lack of support in other operating systems. I.e. no shared Ethernet traffic with z/OS or z/VSE.
- **Shared RDMA traffic (SMC-R) with z/OS works**
- **VEPA mode available starting with with Linux kernel 5.1**

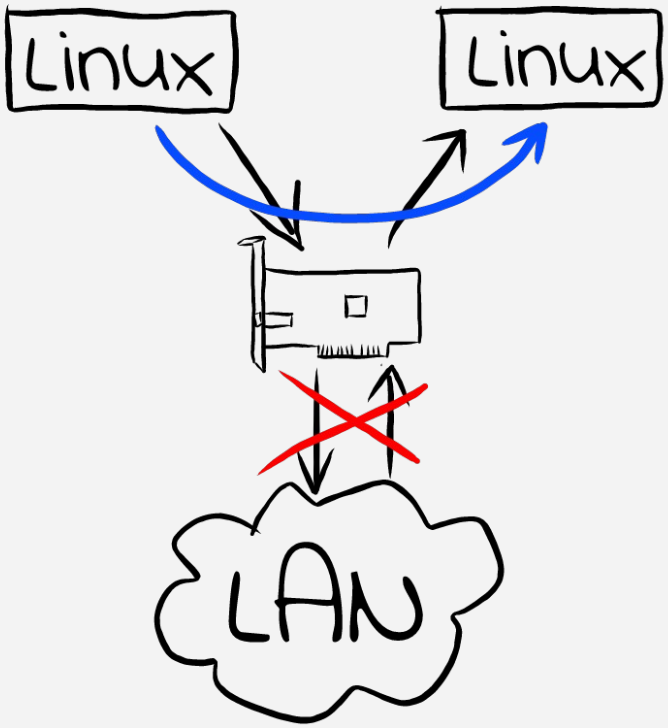



**Fig.1**: *Flow of shared traffic – external traffic will go to the LAN as usual*

# z/VM Support

- **RoCE only supported as passthrough device:**
  - Attach PCI FID to Linux guest:

```
#CP ATTACH PCIFUNCTION <FID> to <guest>
```

  - Configure in guest just like in LPAR case (includes channel bonding)
- **SSI Usage**:
  - Singleton domain only
  - I.e. not supported for live guest migration
- **RoCE not supported in VSWITCH or z/VM's TCP/IP guests – both support OSA exclusively**

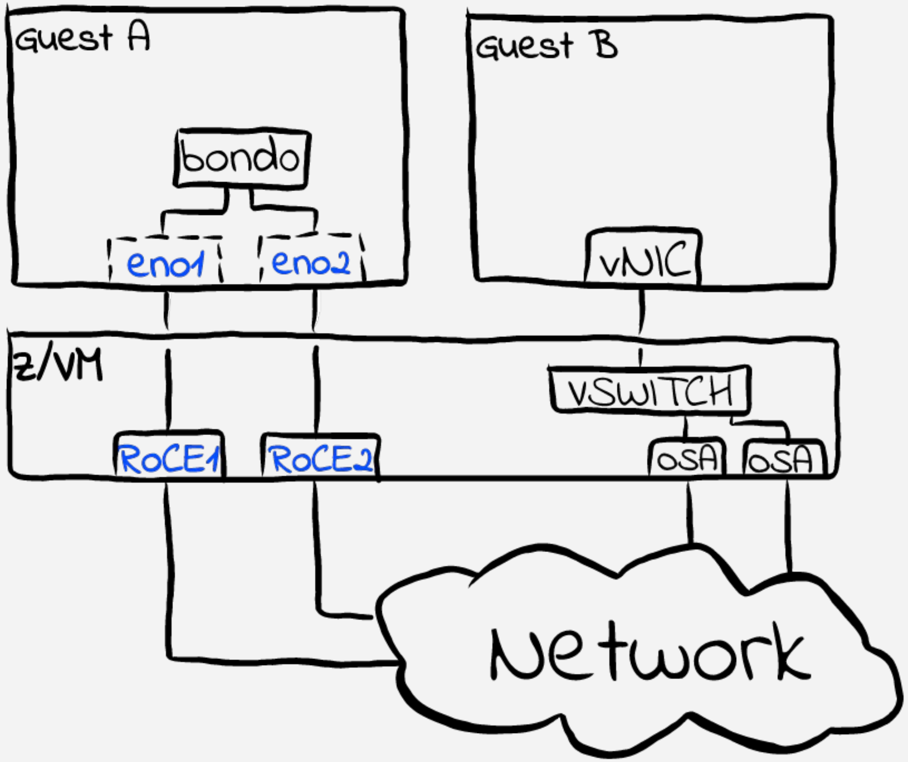



**Fig.1**: *Usage of RoCE Express adapters within z/VM*

# KVM Support

- **Host**
  - No support for Open vSwitch (lack of promiscuous mode)
  - Use RoCE Express adapters for MacVTap, `virtio-net`, et al
  - Configure in guest just like in LPAR case (includes channel bonding)
- **Guest**
  - Requires Linux kernel 4.14, QEMU v2.11 and *libvirt* v4.10.
  - Specify passthrough device in domain XML
    - Use `<source>` to specify the source device
    - Use `<zpci>` to influence the PCI address inside the guest
    - Leave attributes in `<address>` alone – will be auto-filled
  - Configure in guest just like in LPAR case (includes channel bonding)
  - Performance improvements pending
  - See here for further details

```
<hostdev mode='subsystem' type='pci'
            managed='yes'>
   <source>
      <address domain='0x2' bus='0x0'
               slot='0x0' function='0x0'/>
   </source>
   <address type='pci' domain='0x02' \
            bus='0x00' slot='0x01' \
            function='0x0'>
      <zpci uid='0x0001' fid='0x00000000'/>
   </address>
</hostdev>
```

**Fig.1**: *Sample domain XML for PCI passthrough device*

# RDMA Usage

- **SMC-R**
  - <u>**S**</u>hared <u>**M**</u>emory <u>**C**</u>ommunications <u>**R**</u>emote
  - Provides low-latency, high throughput connectivity between CPCs based on RoCE Express
  - Built-in failover for high availability
  - Compatible with z/OS
- **NVMeoF**
- **rdma-core**
- **uDAPL**
  - <u>**U**</u>ser <u>**D**</u>irect <u>**A**</u>ccess <u>**P**</u>rogramming <u>**L**</u>ibrary
  - Allows RDMA traffic directly from/to userspace
  - However: Project is deprecated

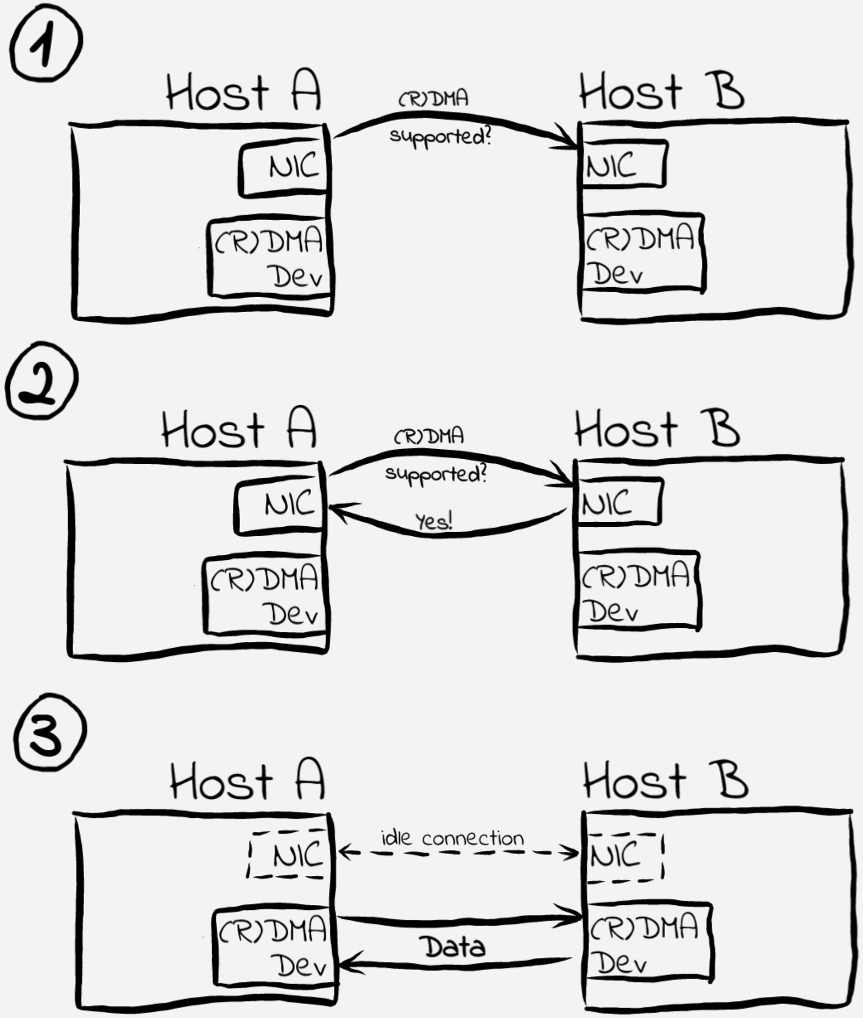
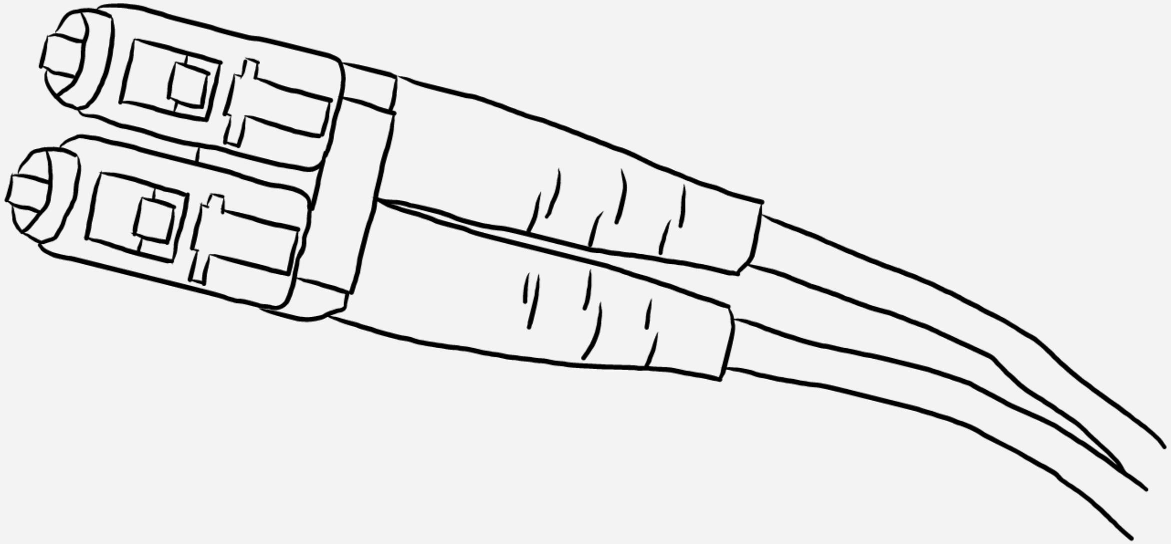



**Fig.1**: *Schematic SMC-R connection setup*

# Agenda

- The Cards
- Device Drivers, Features and Commands
- Usage
- Performance
- Summary
- References



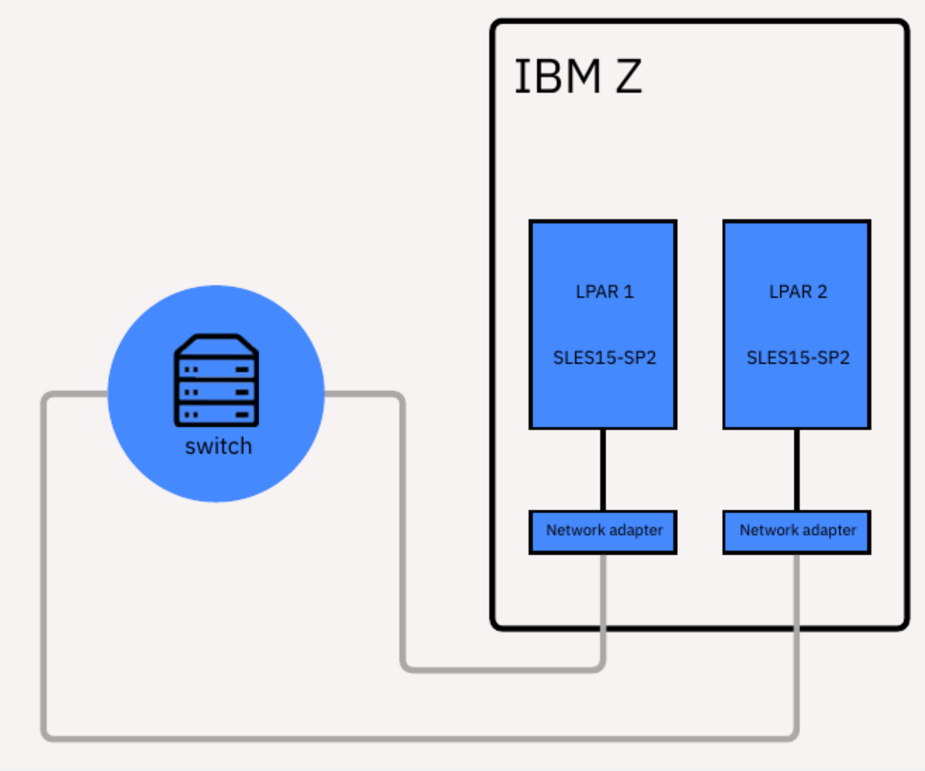
# Setup

## IBM z15

- **Systems: 2 LPARs (client & server), each with**
  - cores: 4 with SMT-2, i.e. 8 threads,
  - memory: 16GB
  - distribution: SUSE Linux Enterprise Server 15 SP2
  - kernel version: 5.3.18-22-default
- **Network adapters: Three different adapter types**
  - OSA-Express6S 10 GbE
  - OSA-Express7S 25 GbE
  - 25 GbE RoCE Express2.1

**The adapters are not shared between the LPARs, i.e. both LPARs have different adapters of each type attached.**

**All adapters are connected to a switch.**



**Workload**: rr1c-200x1000
Highly transactional with medium data sizes.

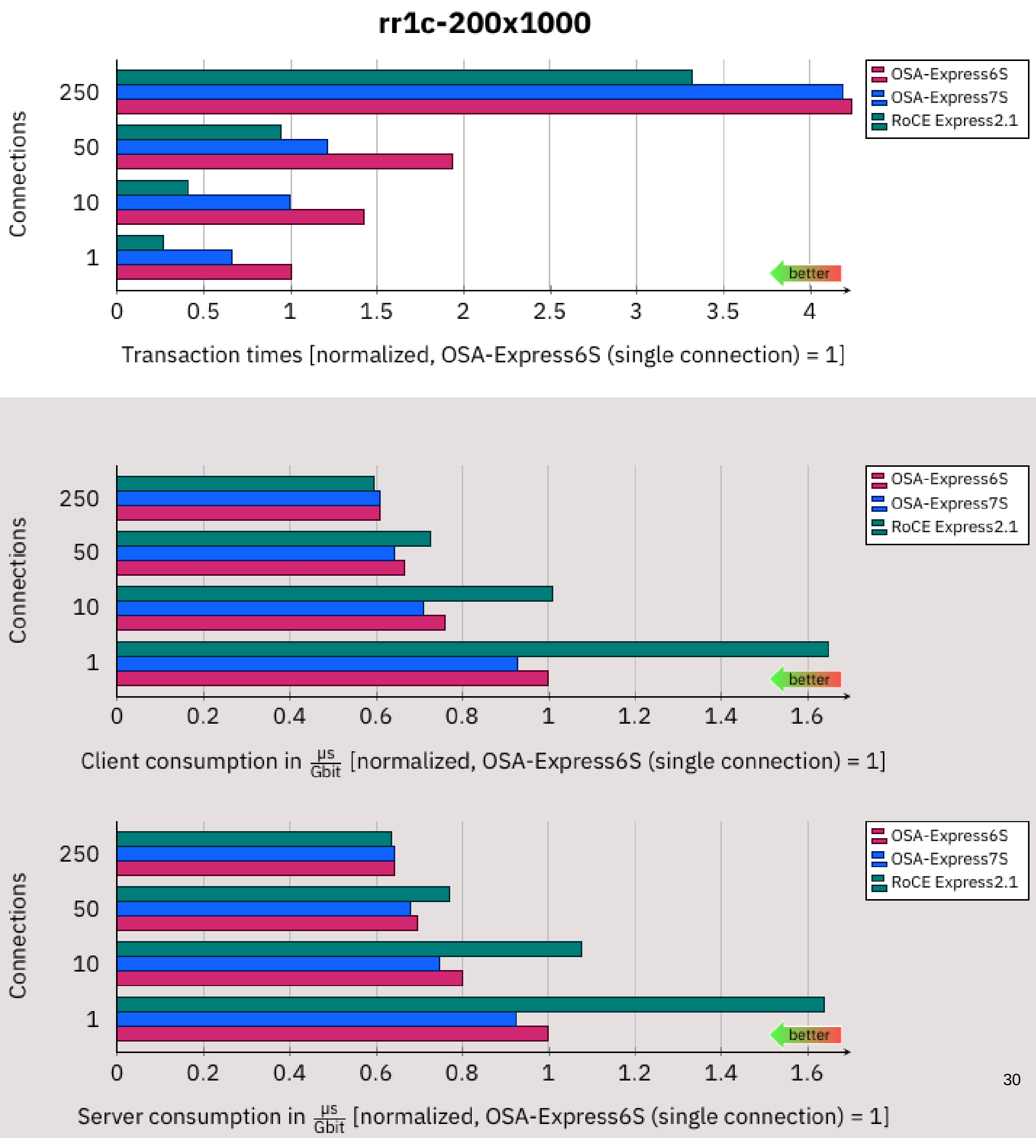
**Note**: Results normalized to OSA-Express6S single connection.

**RoCE Express2.1**: RoCE Express2.1 highly outperforms OSA-Express adapters in speed for this workload pattern.

Transaction time for the single connection case as well as with 10 parallel connections is reduced to less than half the time compared to OSA-Express7S. Even for higher numbers of parallel connections RoCE Express2.1 is significantly faster with a 40% reduced transaction time when running with 50 parallel connections. With 250 parallel connections the improvement is more than 15%.

The improved transaction time comes with a certain cost. Processor consumption (μs/Gbit) is around 75% higher for the single connection case and around 40% higher for 10 parallel connections compared to OSA-Express7S. With 50 parallel connections the gap is closing – around 17% – and for 250 parallel connections RoCE Express2.1 is on the same consumption level as the OSA-Express.

Additional information on this workload – making use of RPS – is presented subsequently.



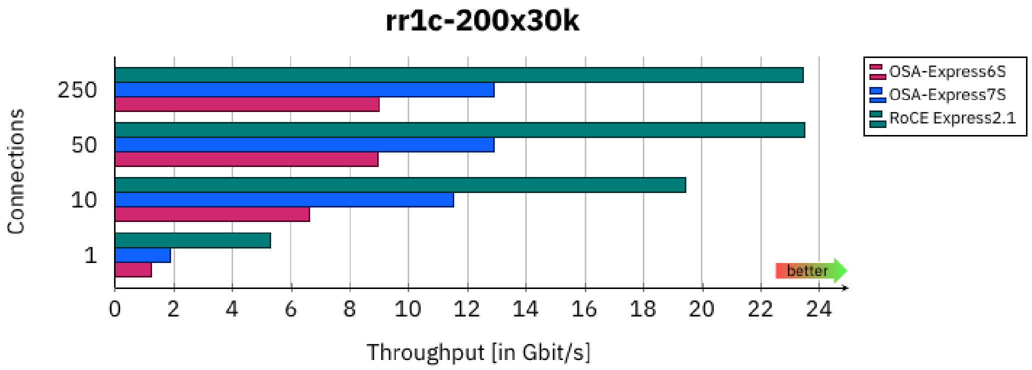
**Workload**: rr1c-200x30k
Transactional with large data size.

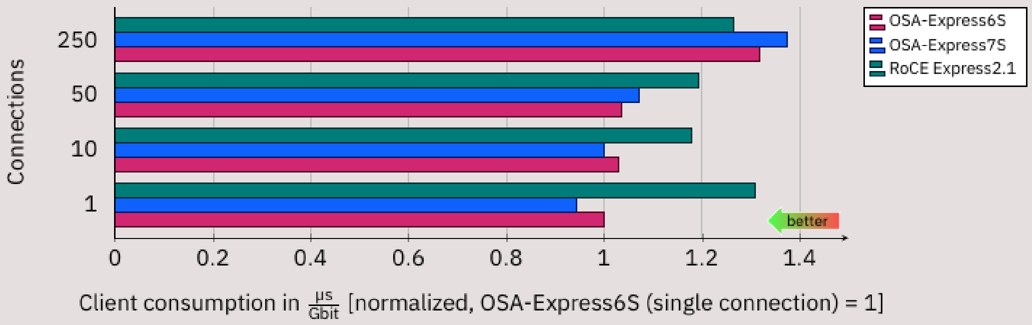
**Note**: Results for processor consumption normalized to OSA-Express6S single connection.

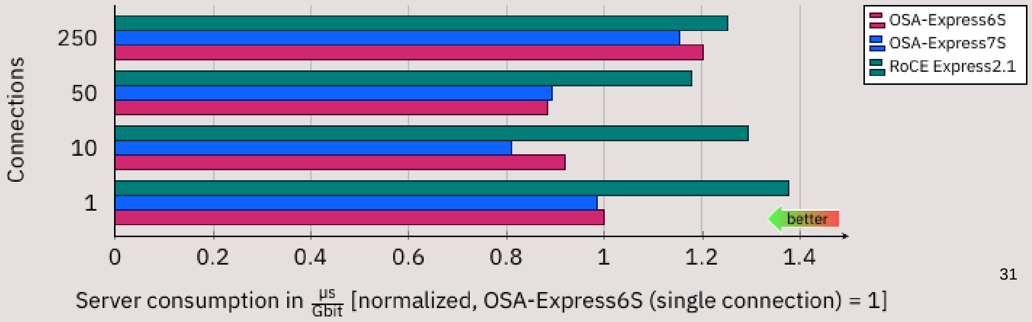
**RoCE Express2.1**: With regard to throughput the RoCE Express2.1 highly outperforms the OSA-Express.

For the single connection workload RoCE Express2.1 increases throughput by 2.8x compared to OSA-Express7S. Even for multi-connection workloads the improvement is 65% (10 parallel connections) up to 80% (50 and 250 parallel connections).

Furthermore, when running with 50 or 250 parallel connections the throughput is capped by line speed. Again this improvement comes at a certain cost in processor consumption. For single and 10 parallel connections the ratio µs/Gbit is 40% to 60% higher compared to OSA-Express7S. However, for 250 parallel connections RoCE Express2.1 is roughly on the same level as OSA-Express.



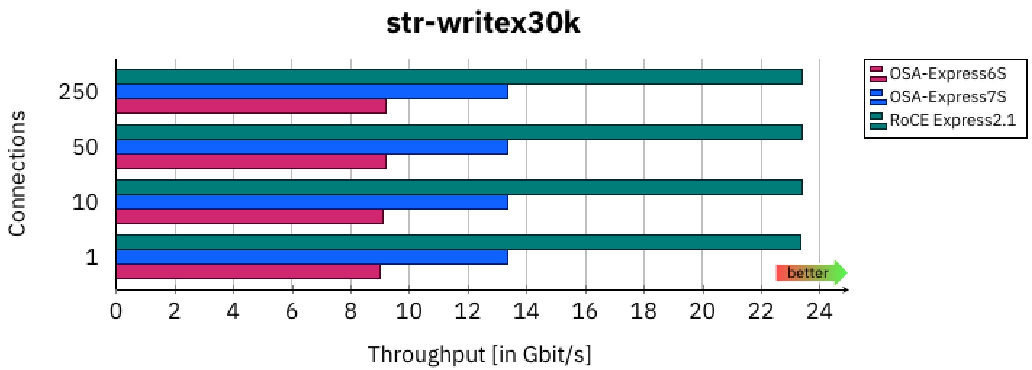



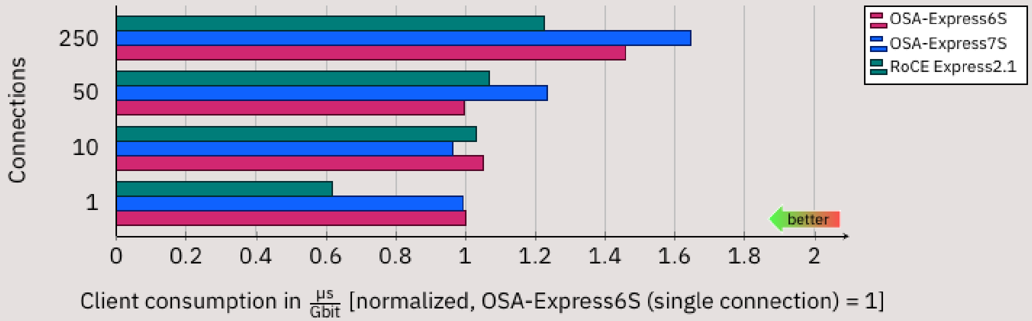
**Workload**: str-writex30k
Streaming workload.

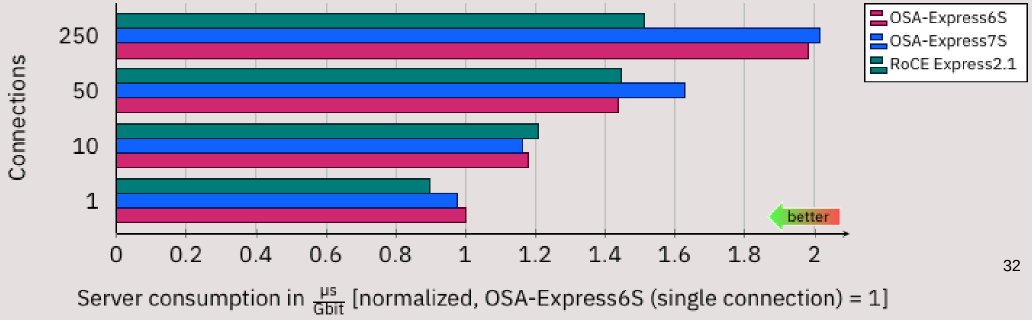
**Note**: Results for processor consumption normalized to OSA-Express6S single connection.

**RoCE Express2.1**: The RoCE Express2.1 reaches line speed in all tested scenarios running this streaming pattern. Compared to OSA-Express7S this is an improvement of around 75%.

In contrast to the request-response workloads, this improvement does not come with an additional cost in processor consumption. For 250 parallel connections, the RoCE Express2.1 is even showing a reduced μs/Gbit ratio on receiver and sender side compared to OSA-Express7S.



# Summary:
# Benefits of PCI Networking Devices with LoZ

**PCI-based Networking Devices make for a better user experience in the Linux on Z ecosystem:**

- Superior performance
- Smoother transition from other platforms, especially x86
- Eliminates need for Z-specific tooling
- Good integration with Linux on Z distributions
- Virtualization via SR-IOV
  ⇒ ***Better integration with Linux on Z ecosystem***

# References

- **Networking with RoCE Express**: The complete guide to everything RoCE on IBM zSeries
  https://ibm.biz/BdPFbT
- **PCIe RoCE Express network guide**: How to install RoCE Express on Linux on IBM zSeries
  https://ibm.biz/BdPFb9
- **Performance of Linux on Z Networking Adapters**
  https://ibm.biz/BdPFbw
- **Linux on Z Documentation**
  https://www.ibm.com/docs/en/linux-on-systems?topic=linux-z-linuxone
- **Webcasts**
  http://ibm.biz/Linux-on-IBMZ-LinuxONE-Webcasts
- **Solution assurance**
  https://ibm.biz/BdPFbk
- **Blogs**
  - **Linux On Z Distributions News**
    https://linuxmain.blogspot.com/
  - **Linux On Z Latest Development News**
    https://linux-on-z.blogspot.com/
  - **KVM on Z**
    https://kvmonz.blogspot.com/

**Trademarks**: See https://www.ibm.com/legal/copytrade for a list of trademarks

# Backup

| | OSA-Express | RoCE Express | SMC-R |
|---|---|---|---|
| **Type** | NIC | NIC | Protocol/Supplementary |
| **Traffic supported** | All | All | TCP same subnet only |
| **Ports** | 1-2 | 2 | - |
| **RAS Support** | Very good | Good | - |
| **Virtualization Capabilities** | Very good | Mode-dependent | - |
| **Latency** | Low | Very low | Very low[1] |
| **Shared Traffic / Support** | Limited / To-From any OS | Very good / To-From Linux only | Very good / To/From Linux or z/OS |
| **CPU Usage** | Low | Varies | Very low |
| **High Availability** | Bonding required | Bonding required | Built-in |
| **z/OS Connectivity** | Yes | No | Yes |
| **Virtual Switch Support** | Yes | No | - |
| **Container Support** | Yes | Yes | No |
| **DPM Support** | Yes | Yes | Yes[2] |

**[1]** Not for short-lived connections **[2]** Configure PNET ID via `smc_pnet` utility

# Linux on IBM zSystems Networking with RoCE Express

Stefan Raspl

Linux & Virtualization on IBM zSystems