vfio-ap: The Perils of the Weird

Antony Krowiak, Pierre Morel, **Halil Pasic**
IBM Corporation

Trademarks

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

AIX*	DB2*	HiperSockets*	MQSeries*	PowerHA*	RMF	System z*	zEnterprise*	z/VM*
BladeCenter*	DFSMS	HyperSwap	NetView*	PR/SM	Smarter Planet*	System z10*	z10	z/VSE*
CICS*	EASY Tier	IMS	OMEGAMON*	PureSystems	Storwize*	Tivoli*	z10 EC	
Cognos*	FICON*	InfiniBand*	Parallel Sysplex*	Rational*	System Storage*	WebSphere*	z/OS*	
DataPower*	GDPS*	Lotus*	POWER7*	RACF*	System x*	XIV*		

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, othr countries, or both.
OpenStack is a trademark of OpenStack LLC. The OpenStack trademark policy is available on the OpenStack website.
TEALEAF is a registered trademark of Tealeaf, an IBM Company.
Windows Server and the Windows logo are trademarks of the Microsoft group of countries.
Worklight is a trademark or registered trademark of Worklight, an IBM Company.
UNIX is a registered trademark of The Open Group in the United States and other countries.

* Other product and service names might be trademarks of IBM or other 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 from 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 from 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.

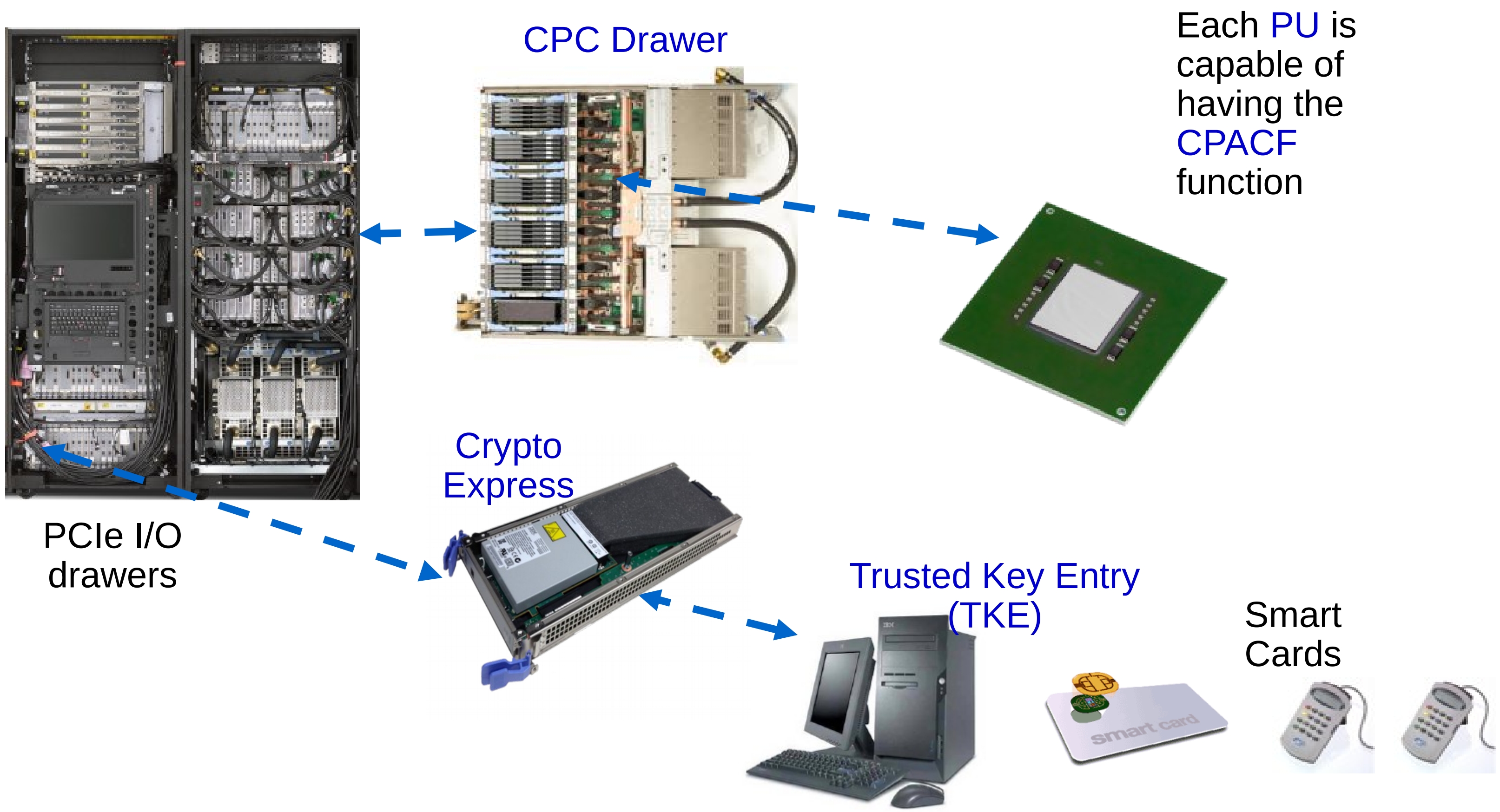
This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g, zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

vfio-ap objective: KVM-based, hardware assisted, pass-trough for AP **Crypto** on IBM z.

Why should anybody care about AP Crypto?

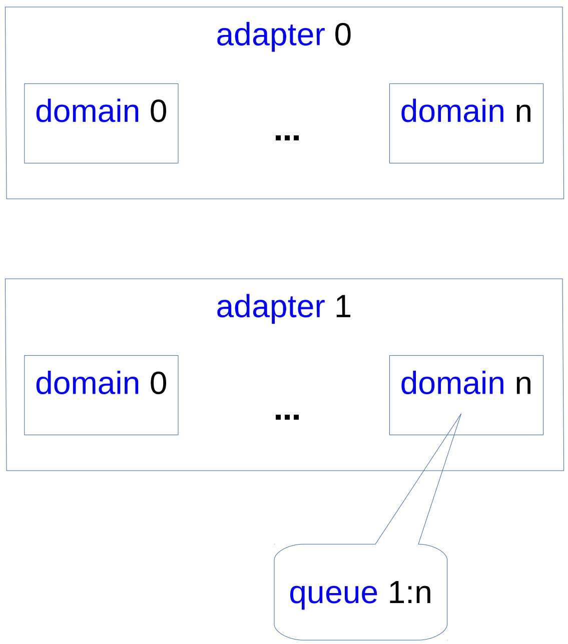
- Adjunct Processors, a.k.a. Crypto Express Features: crypto cards (PCIe)
- Cool because:
  - **Tamper-sensing, tamper-responding HSMs**
  - Secure and protected keys
  - Configurable – 3 different FW loads:
    EP11, CCA, Accelerator
  - Certification (e.g. CEX6C and CEX6P **FIPS 140-2, Level 4**)
- Complementary to CPACF
- Designed with virtualization in mind.

Overview – HW Crypto support in IBM Z



Names

- AP == adjunct processor == Crypto Express feature == adapter ; identified by APID
- Each adapter is partitioned into domains; identified by APQI.
- APID + APQI = APQN; identifies an AP queue, which is, from a **functional** perspective, the **device** providing the crypto services, e.g. HSM.
- The functionality is made available to SW via 3 instructions: NQAP, DQAP, PQAP
- NQAP and DQAP act strictly on an AP queue
- PQAP is somewhat special (config info, resets, etc)



IBM z – Where everything is virtualized

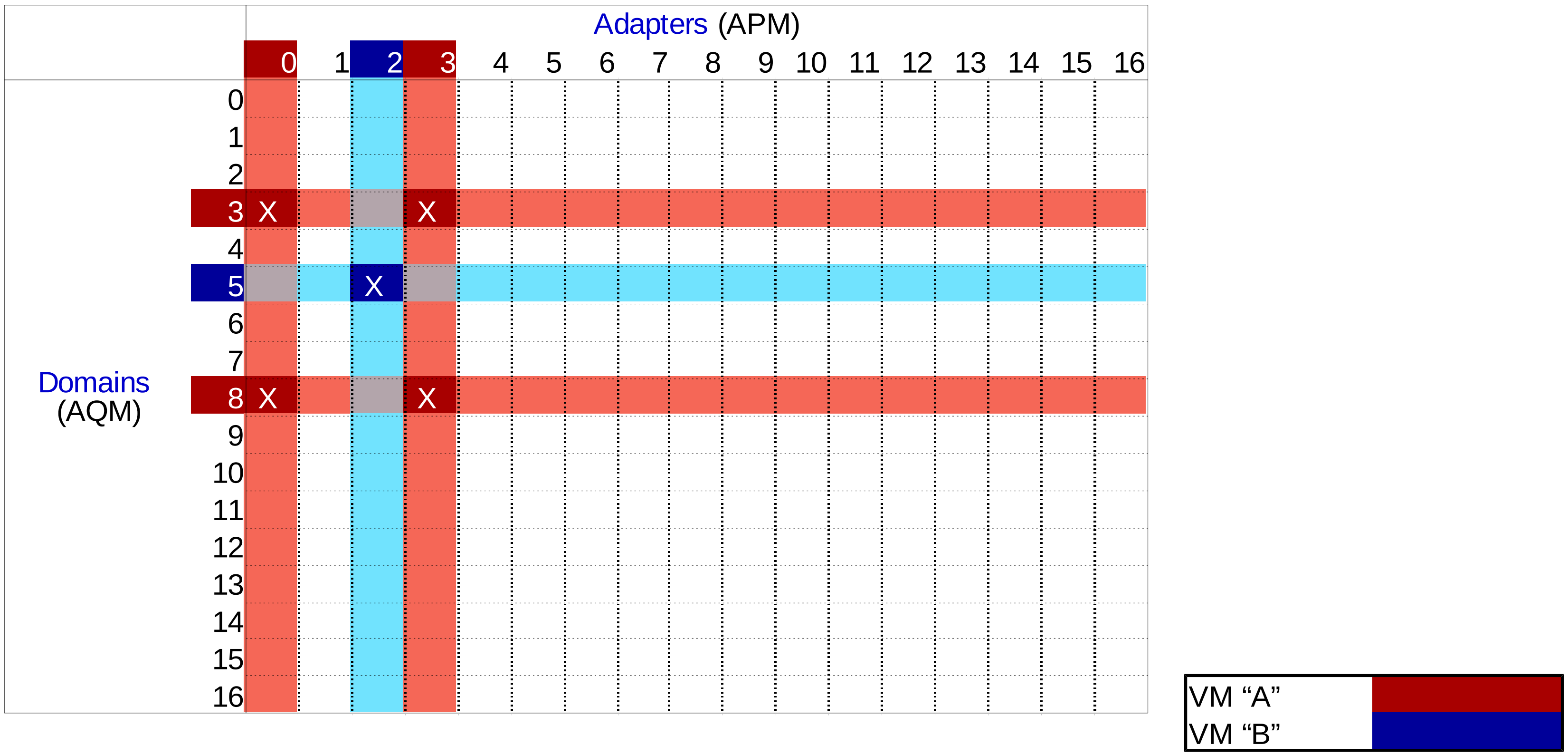
- Big Machines! Only FW is allowed to run 'native-native'. Customer workload can be:
  - LPAR: Logical Partition, the 'new native' (G1)
  - KVM guest (G2)
  - Nested virtualization (Gn, 2 < n < 8?)

- The SIE instruction
  - Execute a vCPU based on several control structures in host storage (memory), i. e. State Description (SD) and SD-satellites.
  - Keep executing the vCPU until:
    - Hypervisor cooperation is needed
    - The hypervisor wants to intervene
    - Stuff happens

Good news! SIE's AP virtualization scales beyond G1

- Remember LPAR is the **new native**, which is already virtualized. At LPAR level, the story is mostly about **partitioning resources**.
- AP resources are **partitioned** in the following way. Each LPAR has three masks in an SD-satellite that control access to AP queues:
  - APM: if bit corresponding to the adapter not set, the guest can do nothing with the adapter
  - AQM: if bit not set the guest can not ***use*** the given domain (on any adapter)
  - ADM: if bit not set the guest can not ***control*** the given domain (on any adapter)
  - The Cartesian product:
    - APM x AQM: authorizes **AP queue use**
    - APM x ADM: authorizes **AP queue control**
- For G2 (and higher), APM, AQM and ADM are effective controls (i.e. EAPM = G1.APM & G2.APM); so, KVM only needs to **sub-partition** and almost everything works. Per architecture, on each guest level, **full sized masks** are used **regardless of** what is **installed** or made **available** by lower virtualization layers.

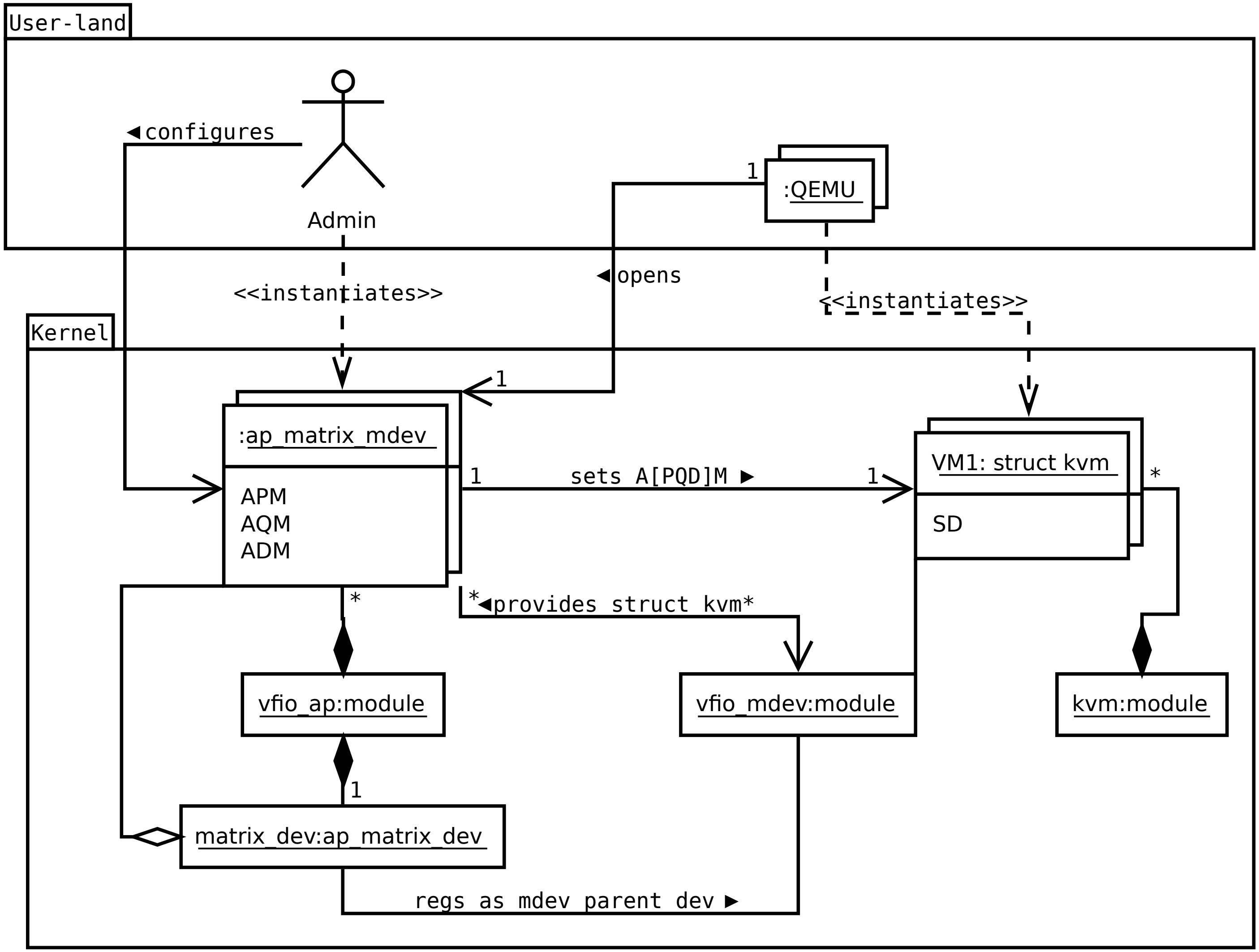
Example – APM, AQM, ADM (APCB)



How do we model this in SW?

- Kernel view: usually, **assignment → vfio**
  - Assign a **full device** with plain **vfio**, or
  - Assign a **uniform part** of a device with **vfio-mdev**
  - Usually, we do not deal with devices that are not available
- QEMU view: usually, model and function in sync
- AP crypto in Linux (host)
  - **Card** devices
  - **Queue** devices: Live within the scope of card devices
  - **Zcrypt device**: Provides crypto for user-land, load balanced over AP's
- We can't/don't want to pass-through:
  - **Queue** devices: **too fine** grained, SIE can't do it
  - **Card** devices: **too coarse** grained
- Design decision: Regard the *whole* **AP subsystem** as one device that is **shared** (mdev) between different **guests** and the **host**.

The grand design



For us vfio-mdev is ...

- … a good match because, we are almost like a normal mediated device:
  - we kind of do have a host device can be **shared** scenario
  - we get a host **device that stands for the passed-through resources** (for QEMU)
  - we get a pointer to struct kvm to do our virtualization stuff
- … not a perfect match because:
  - we deeply **care about what queues** are assigned to what entity (key material)
  - it is **not one size fits all,** like the original mdev design (for vGPUs) implies
    - life-cycle: start empty after **create** and build from there
    - **available_instances is weird** for us
  - there is **no trivial**/suitable **mdev parent device**
  - **sharing of queues is not allowed**, constraints on the partitioning
  - queues **reserved for host** usage **must not be accessible for guests** and vice-versa, however the admin should decide what is reserved for host
    - not even if **device flickers**
  - we should be able to authorize (assign) queues that are not yet known to the system (system architecture vs mdev architecture)

Enforcing constraints

- Queues used by (host) zcrypt vs ‘alternative driver’
  - ap_bus got it’s own APM and AQM called apmask and aqmask respectively; can be set via sysfs or via kernel cmd line
  - zcrypt queue drivers bind only to what is specified by the masks, alternative drivers bind only to the complement (vfio-ap is the only alternative driver)

- On each assign_adapter and assign_domain we check whether the resulting queues are:
  - Bound to the vfio_ap driver
  - Not claimed by another vfio_ap_mdev

Life cycle

1) Take care of ap_bus, vfio_ap module

2) Create vfio_ap mdev device:

```
$ uuid=$(uuidgen)
$ echo ${uuid} > /sys/devices/vfio_ap/matrix/mdev_supported_types/vfio_ap-passthrough/create
```

3) Assign resources to the mdev device

```
$ echo 04 > /sys/bus/mdev/devices/${uuid}/assign_adapter
$ echo 04 > /sys/bus/mdev/devices/${uuid}/assign_domain
$ echo 04 > /sys/bus/mdev/devices/${uuid}/assign_control_domain
```

4) Include the mdev device into a VM

   1) QEMU cmd line:
      `qemu -device vfio-ap,sysfsdev=/sys/bus/mdev/devices/${uuid}`

   2) **open** on vfio-ap qdev realization hooks up the vfio_ap_mdev with the struct kvm which **makes the vfio_ap_mdev immutable (i.e. no (un|)assign, remove)**

Life cycle challenges 1

- Create
  - **Libvirt** does **not** seem to be **keen** on doing life cycle management of mdev devices, particularly on **tying mdev creation to guest life cycle** events.
  - OTOH we have persistent configurations where **certain elements are mutually exclusive** with regards to full instantiation. For example:
    - Guest1: domain 1; adapters 1, 2
    - Guest2: domains 1, 2 ; adapters 2, 3 conflicts Guest 1 on queue (2,1)
    - Guest3: domain 2; adapter 1 no conflicts (assuming we resolve conflict between G1 and G2)
  - Creating **all mdevs on** system **bring-up** is not optimal.
  - Burdening the **client** of libvirt with **ensuring the vfio_ap_mdev** referenced by the domain **is created before starting the domain** does not seem right *to me* either.
  - Interim solution: Advise against conflicting configs, and **make** create **all on bring-up easy**.

Life cycle challenges 2

- Not yet resources.
  - Currently we only allow resources bound to the vfio_ap driver to be assigned. That is *IMHO* sub-optimal, because we **take away functionality** provided by lower level hypervisor **for no good reason**.
  - Resources may go away, so it isn't an invariant.
- Hot(un|)plug
  - Currently **hot plug is prohibited**, but this is likely to change soon.
  - The assign/unassign interfaces are not best suited for hot plug *IMHO.*
  - The admin *could* make 'alternative' devices 'zcrypt' devices again. React how?
- Migration
  - **CPU model guarded**, yeah!
  - Currently not supported: **vfio-mdev** device (QEMU) is a **migration blocke**r
  - Mighty tricky from technical feasibility perspective.

Outlook

- Hot plug!!
- Life cycle management!

- Clean up?
- Intercept and mediate with address virtualization?
  - Performance vs flexibility.
- Intercept and emulate??
- Migration???

Q&A

Learn more

- Learn about vfio-mdev:
  [2016] vGPU on KVM - A VFIO Based Framework by Neo Jia & Kirti Wankhede
  https://www.youtube.com/watch?v=Xs0TJU_sIPc

- Learn about vfio:
  [2016] An Introduction to PCI Device Assignment with VFIO by Alex Williamson
  https://www.youtube.com/watch?v=WFkdTFTOTpA

- More about vfio-mdev: Check out the Documentation and the doc folders in the Linux kernel and the QEMU source tree respectively.