Live update: persisting IOMMU domains across kexec

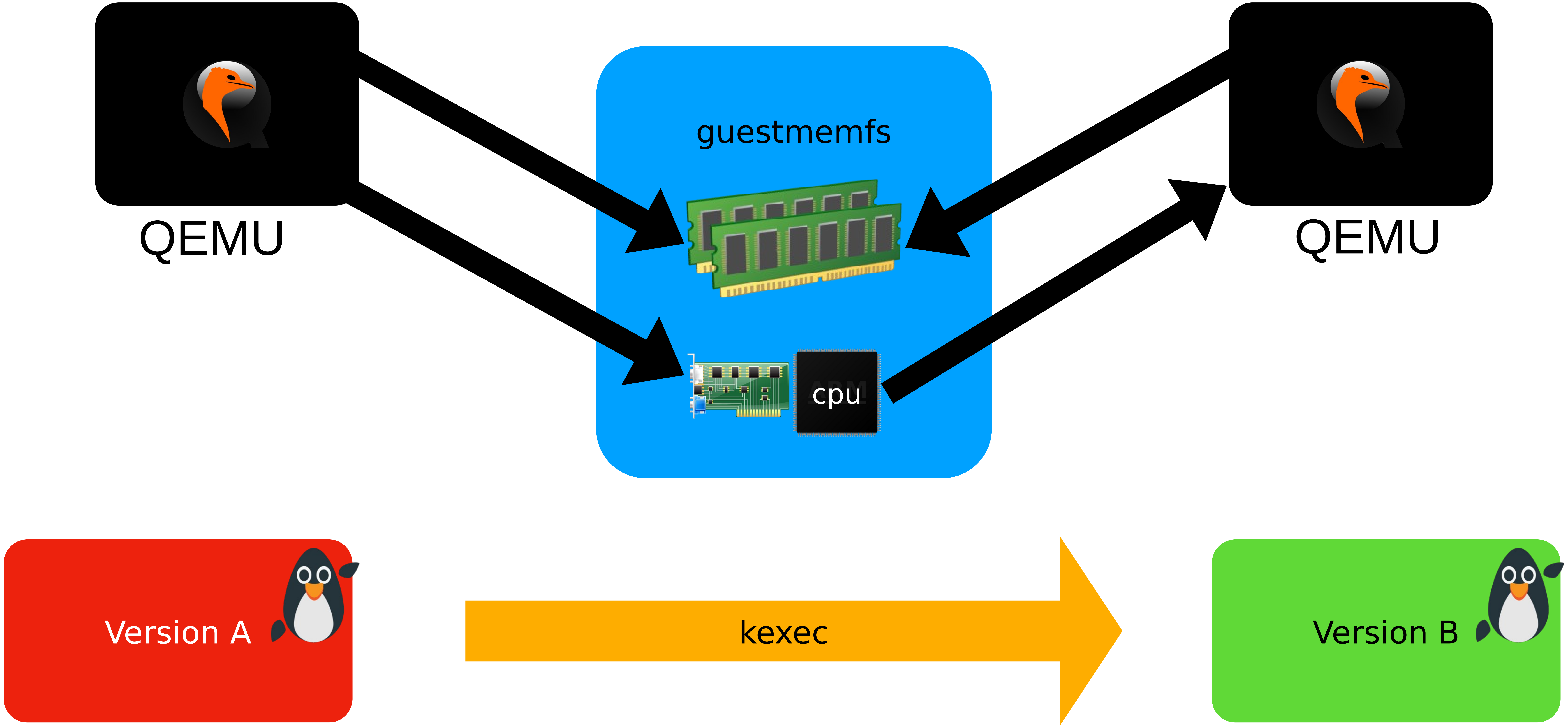
Linux Plumbers Conference, 2024

James Gowans (AWS EC2)
Alex Graf (AWS EC2)

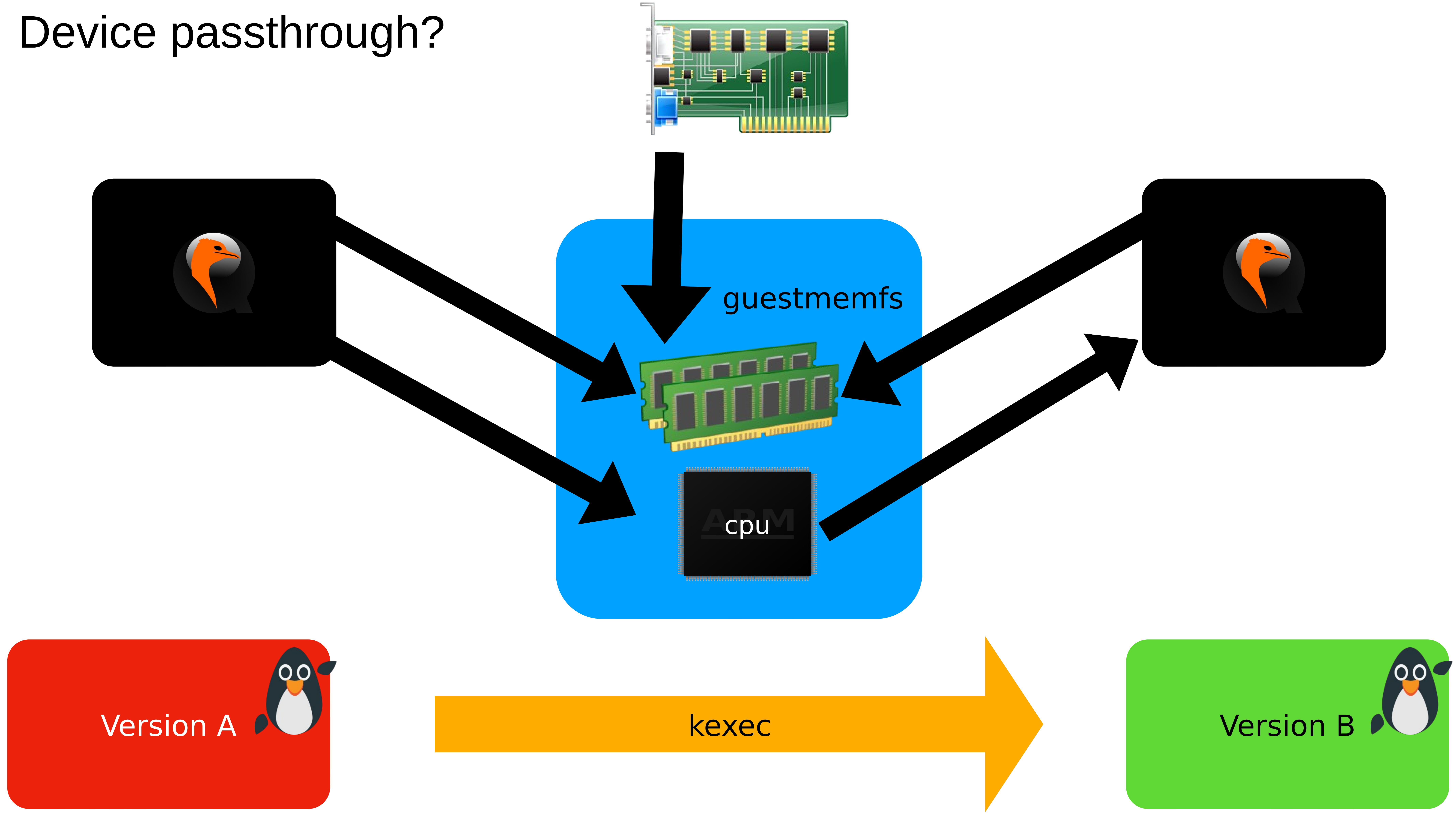
Overview

- Problem overview
- Proposed solution: iommu(fd) persistence via KHO
  - Userspace APIs and internals
- Discussion topics:
  - Are we looking at the problem correctly?
  - Userspace interfaces to solve this?

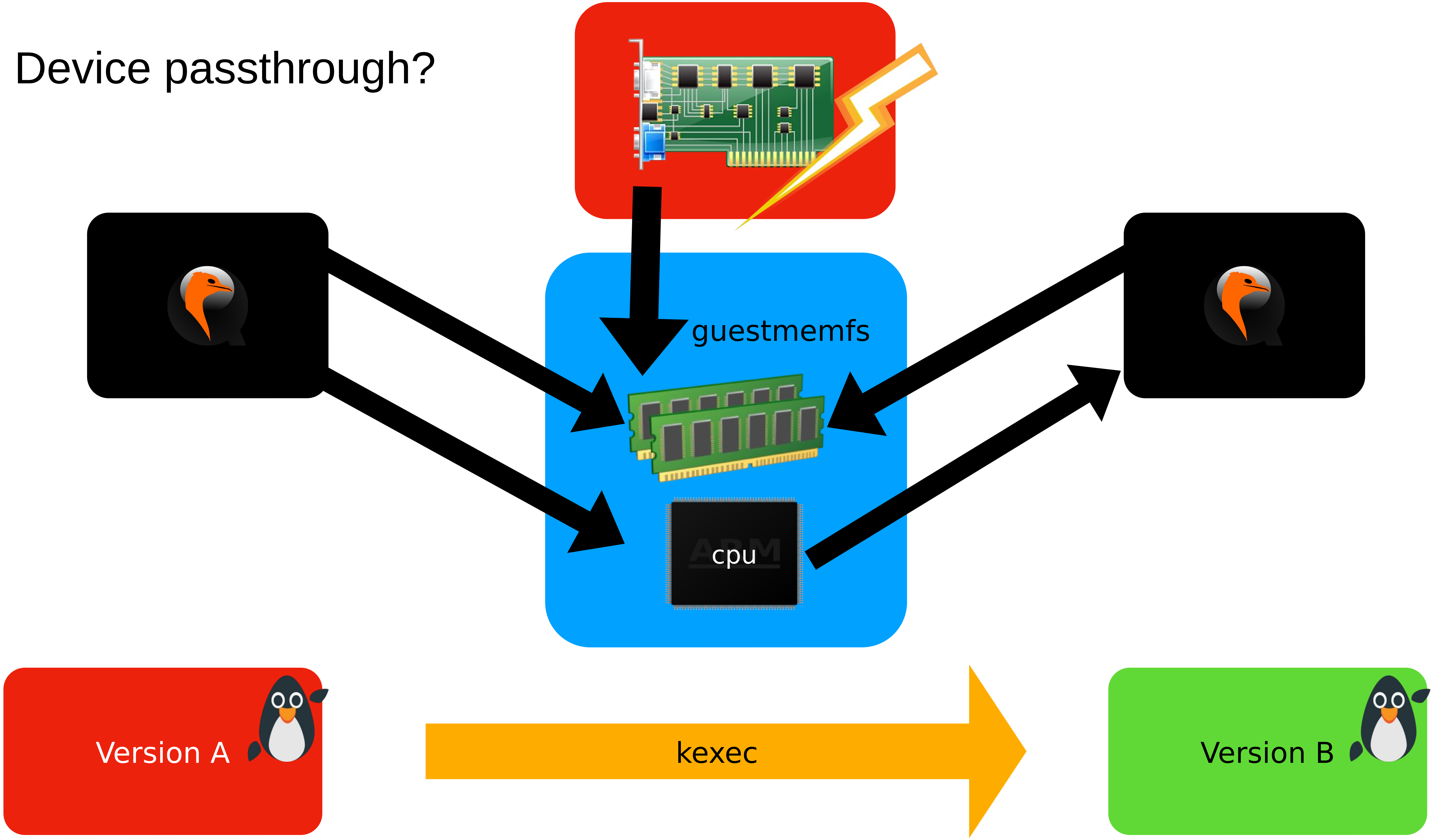
Live update without device passthrough:



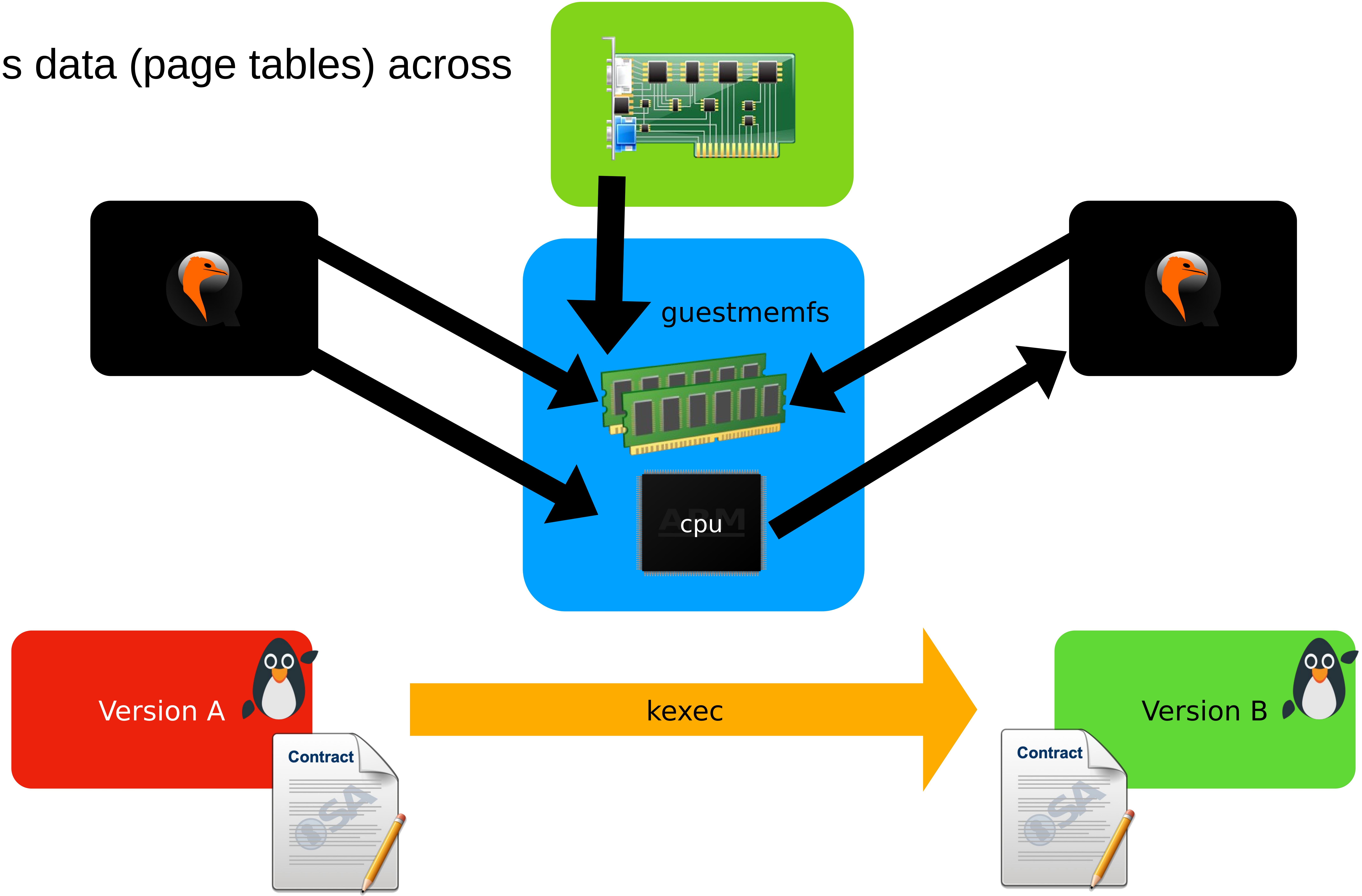
Device passthrough?



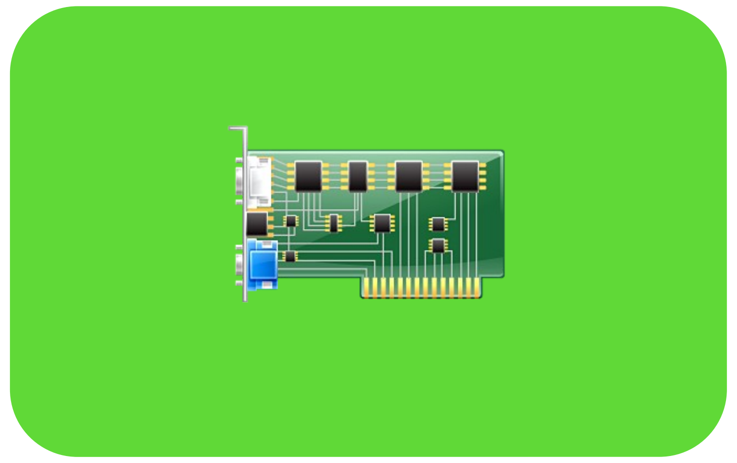
Device passthrough?



Pass data (page tables) across



Roughly:



- Mark IOMMUFD as persistent
- Iommu driver tags iommu_domain persistent
- Serialise struct fields and pgtable pages (via KHO)
- Deserialise and expose to userspace

Mark persistent

```
struct iommu_option option = {
   .option_id = IOMMU_OPTION_PERSISTENT,
   .op = IOMMU_OPTION_OP_SET,
   .val64 = 0 /* output value - persistent ID */
};
```

Now IOAS and HWPT can also be set as persistent.

All HWPT for a persistent IOAS must be persistent

Mapped memory must be persistent too! (guestmemfs)

After kexec:

```
/sys/kernel/persistent_iommufd/<id>/iommufd
```

How?

Kexec Handover “KHO” framework:

https://lore.kernel.org/all/20240117144704.602-1-graf@amazon.com/#r

Device driver / module serialise callbacks

Device tree blob for fields and memory pages

Iommufd serialise descriptor

After kexec: grab state out of KHO

IOMMU driver mark pgtable as persistent

Keep IOMMU enabled, only zap non-persistent pgtables.

```
        intel-iommu {
                domains {
                        1 {
                                Mem = [ …… pgtable pages ….. ];
                                persistent_id = <0x1000000 0x00>;
                                pgd = <0xa0eb27 0x1000000>;
                                agaw = <0x1000000>;
                                devices {
                                        0 {
                                                bus = [00];
                                                devfn = [10];
                                        };
...

        iommufd {
                iommufds {
                        1 {
                                ioases {
                                        2 {
                                                pinned-file-handle = <0x00 0x00>;
                                                0 {
                                                        iova-start = <0xc00 0x00>;
                                                        iova-len = <0x800200 0x00>;
                                                        iommu-prot = <0x5000000>;
                                                };
...
```

Discuss

Looking at problem correctly?

Userspace interfaces?

KHO as transport layer?

RFC to sketch out problem/solution:

https://lore.kernel.org/all/20240916113102.710522-1-jgowans@amazon.com/