Persisting guest memory and kernel/device state safely across kexec

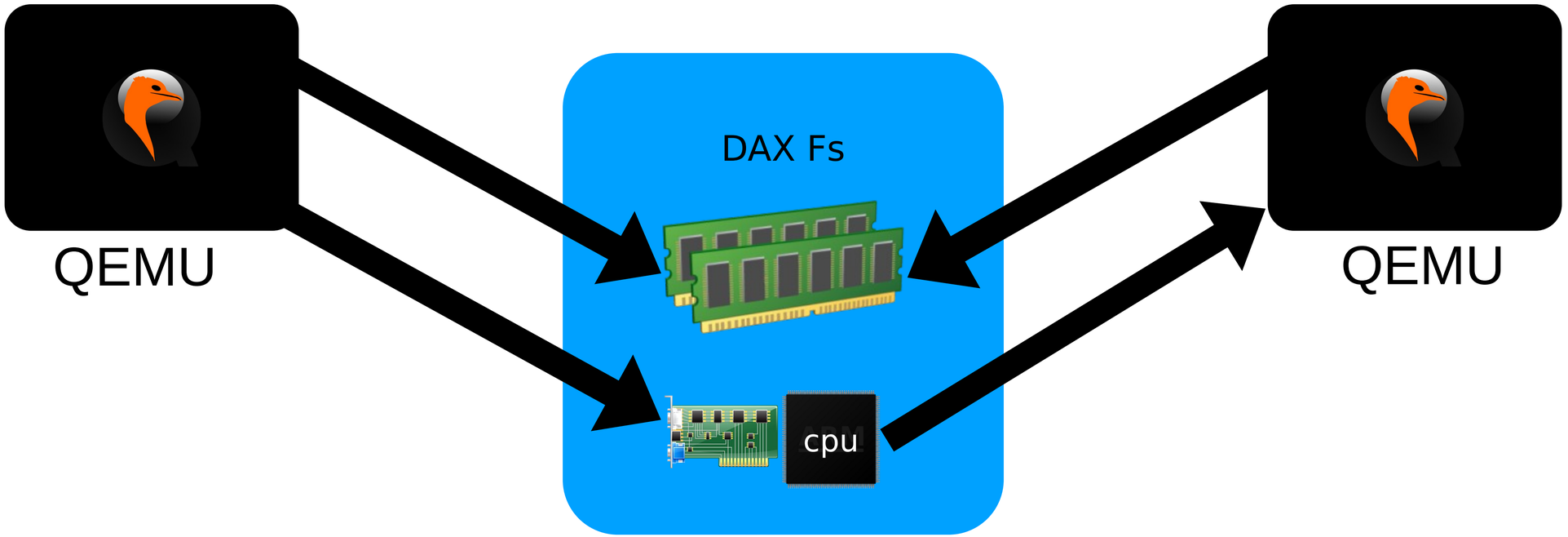
Linux Plumbers Conference, 2023

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

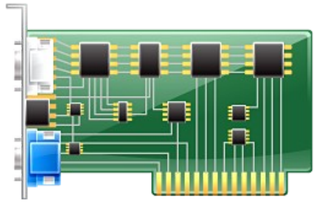
Overview

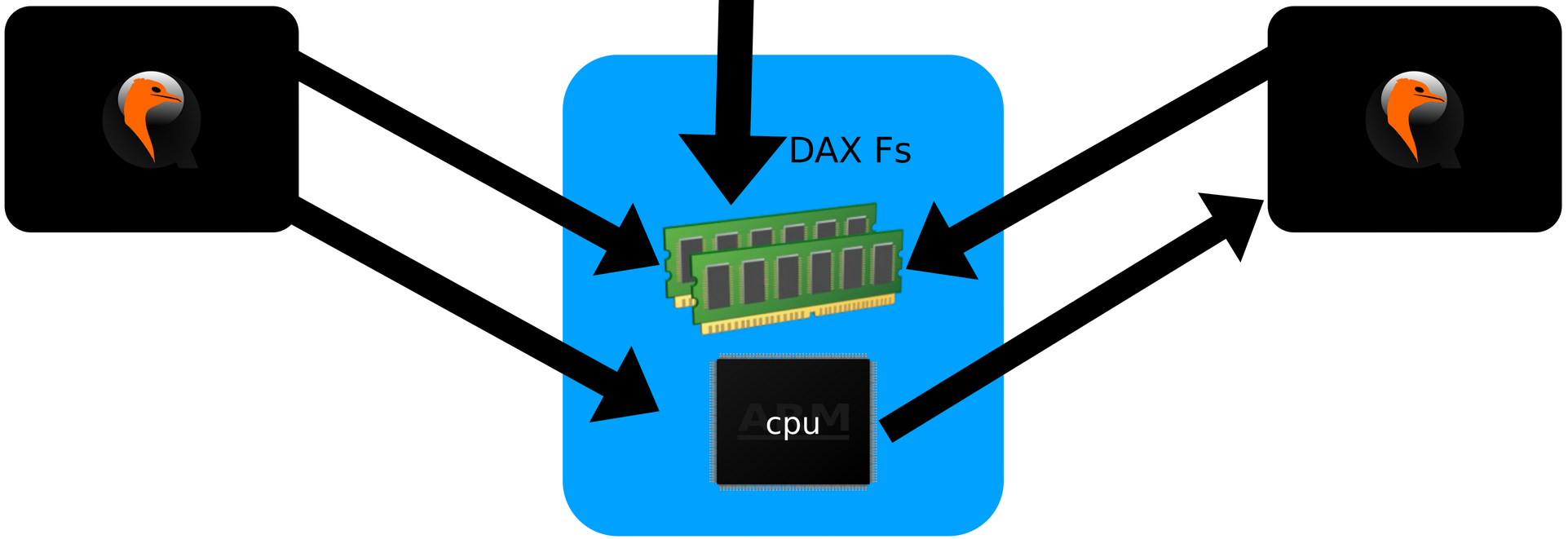
- Live update, what we can do today
- What we can’t do today
- Possible ways to solve it
- Discussion topics!
    - How should we solve this?

Live update today:



Device passthrough?



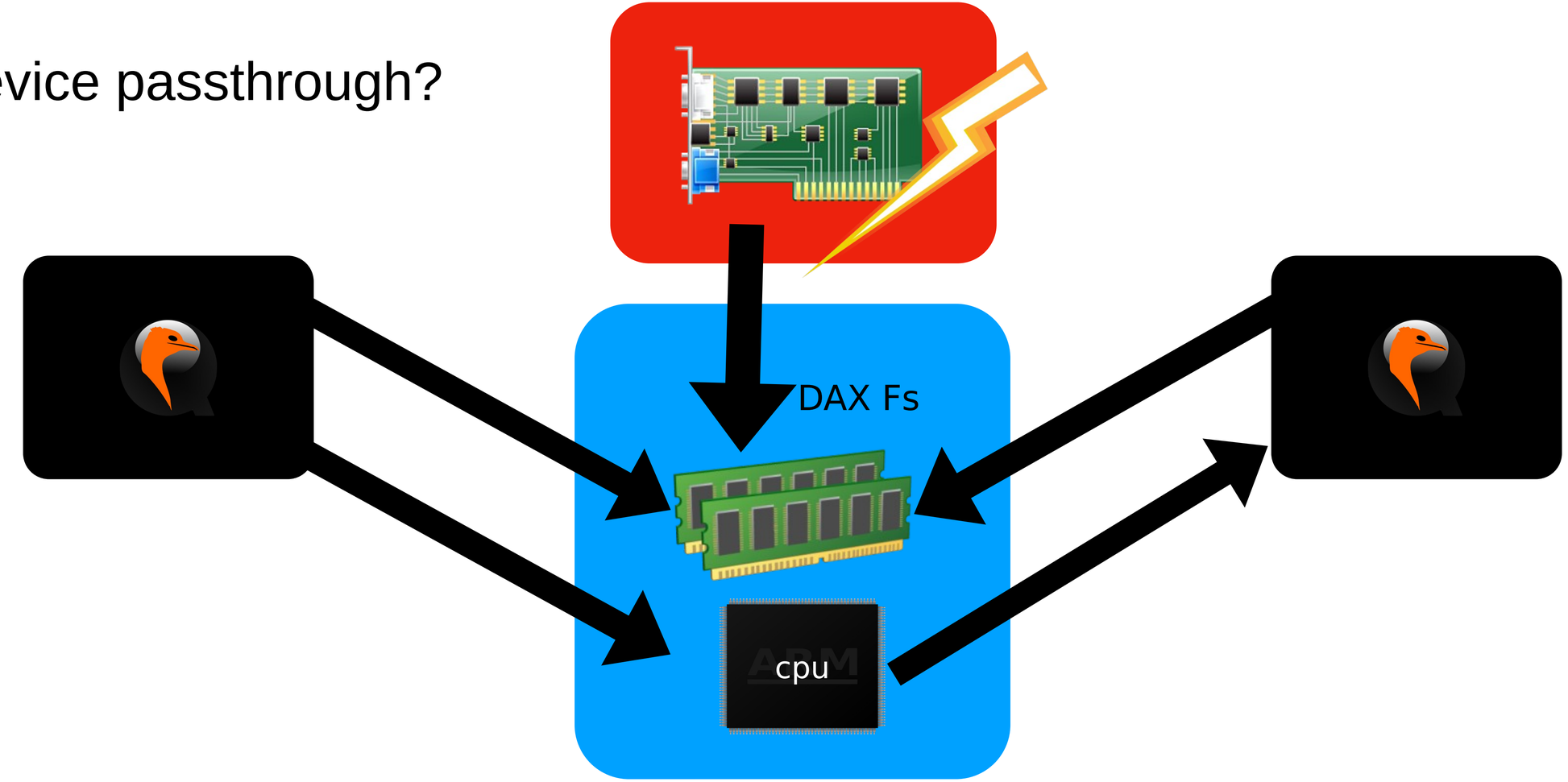


Version A

kexec

Version B

Device passthrough?

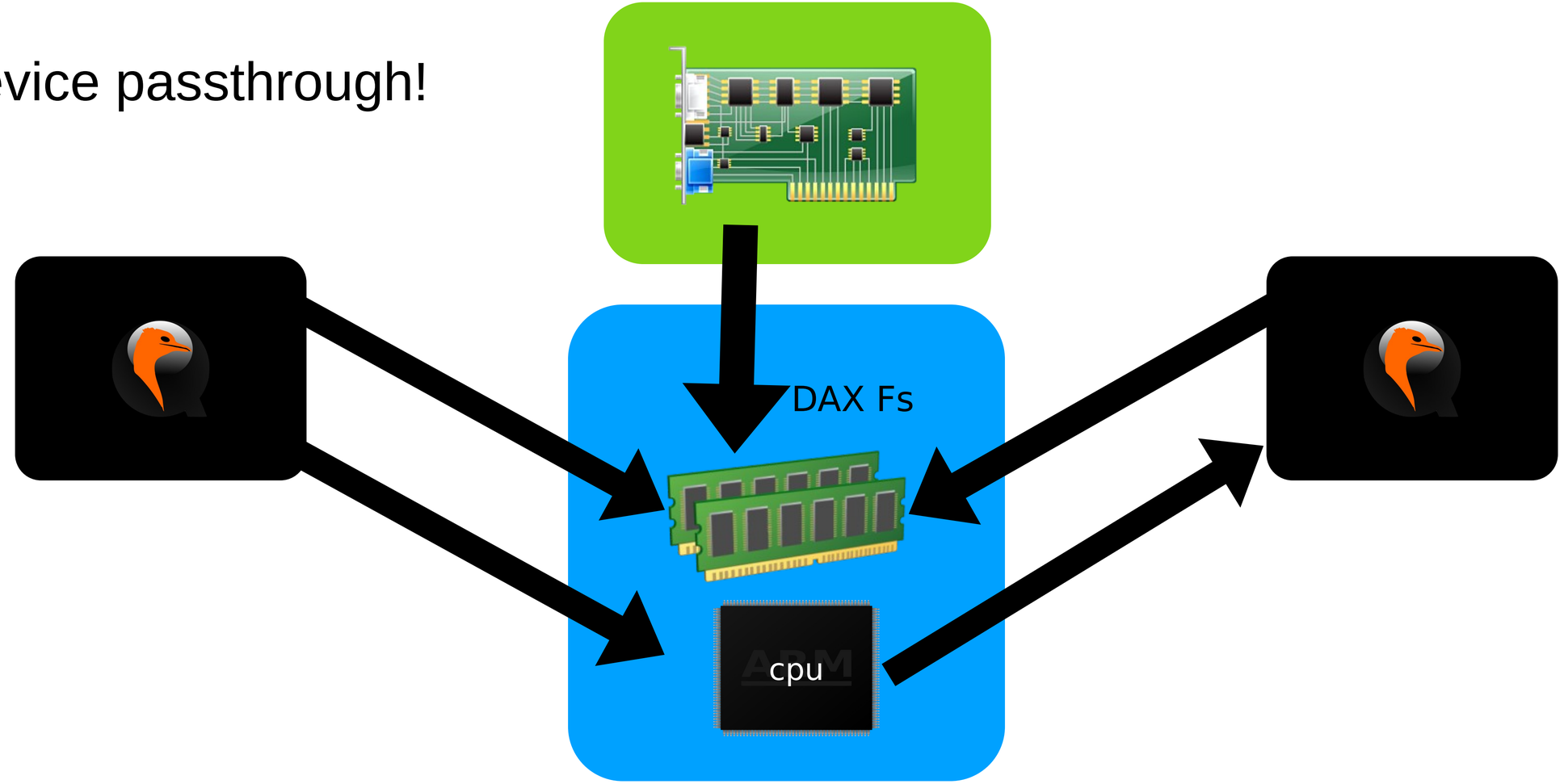


Version A

kexec

Version B

Device passthrough!

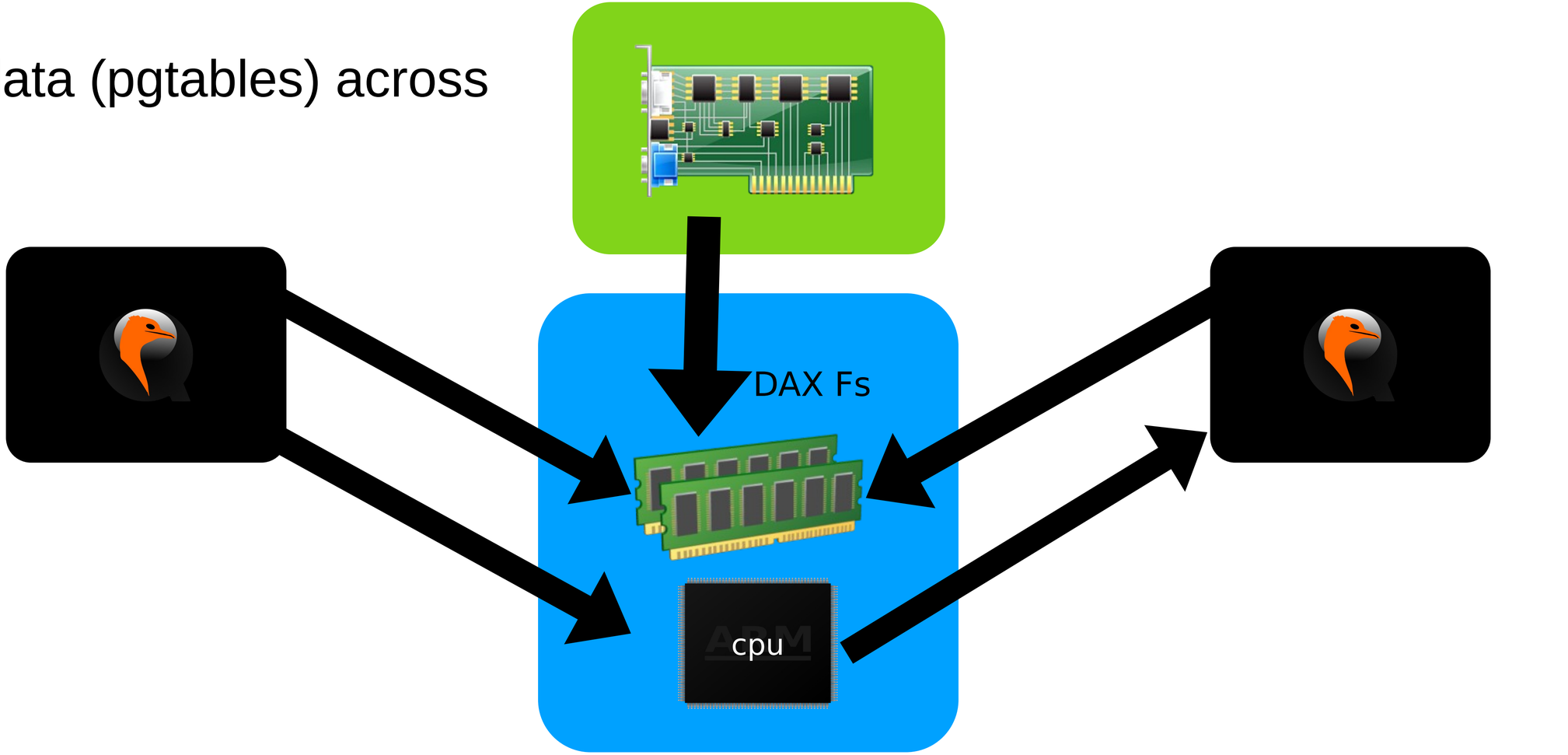


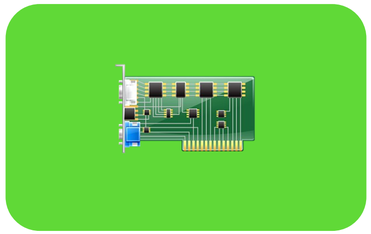
Version A

kexec

Version B

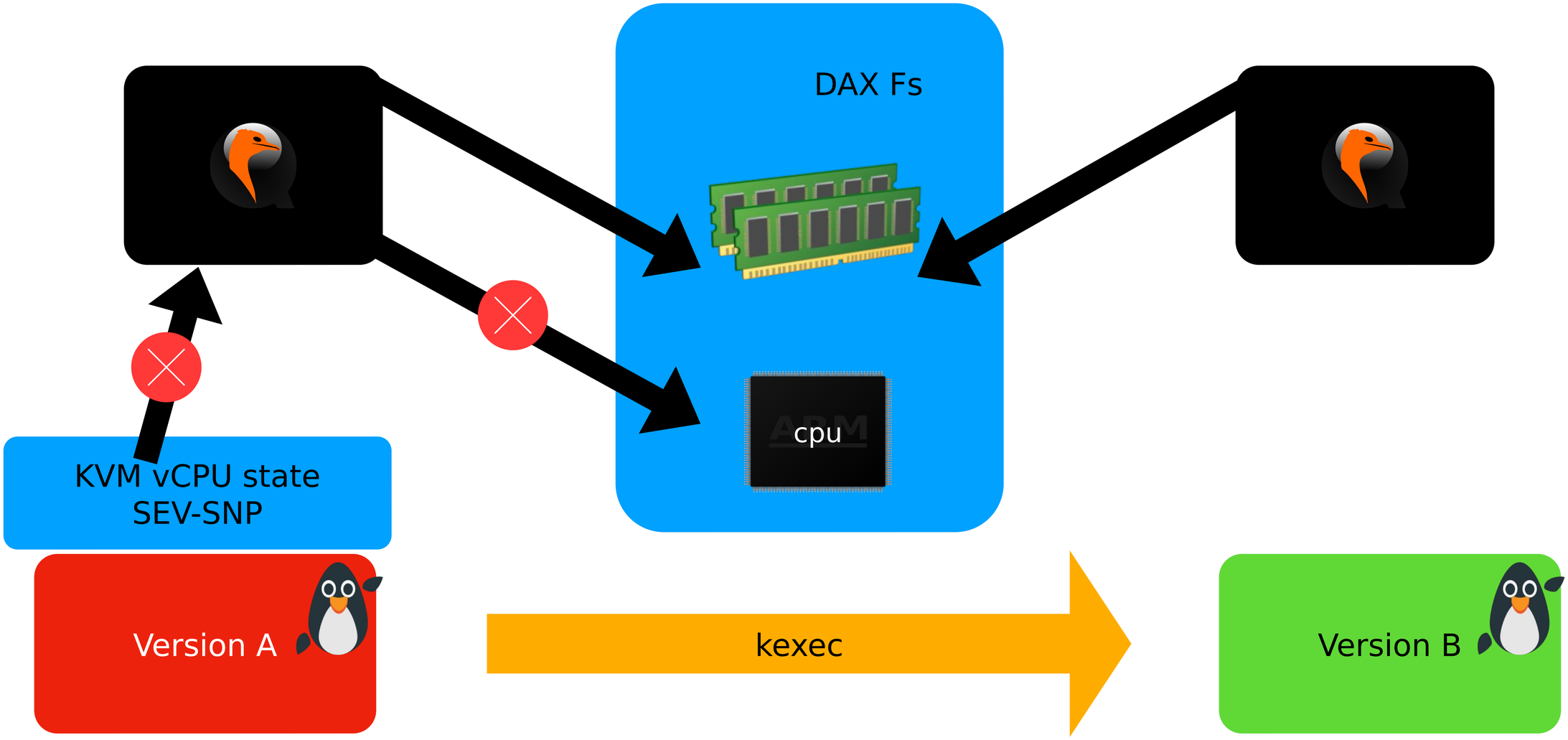
Pass data (pgtables) across



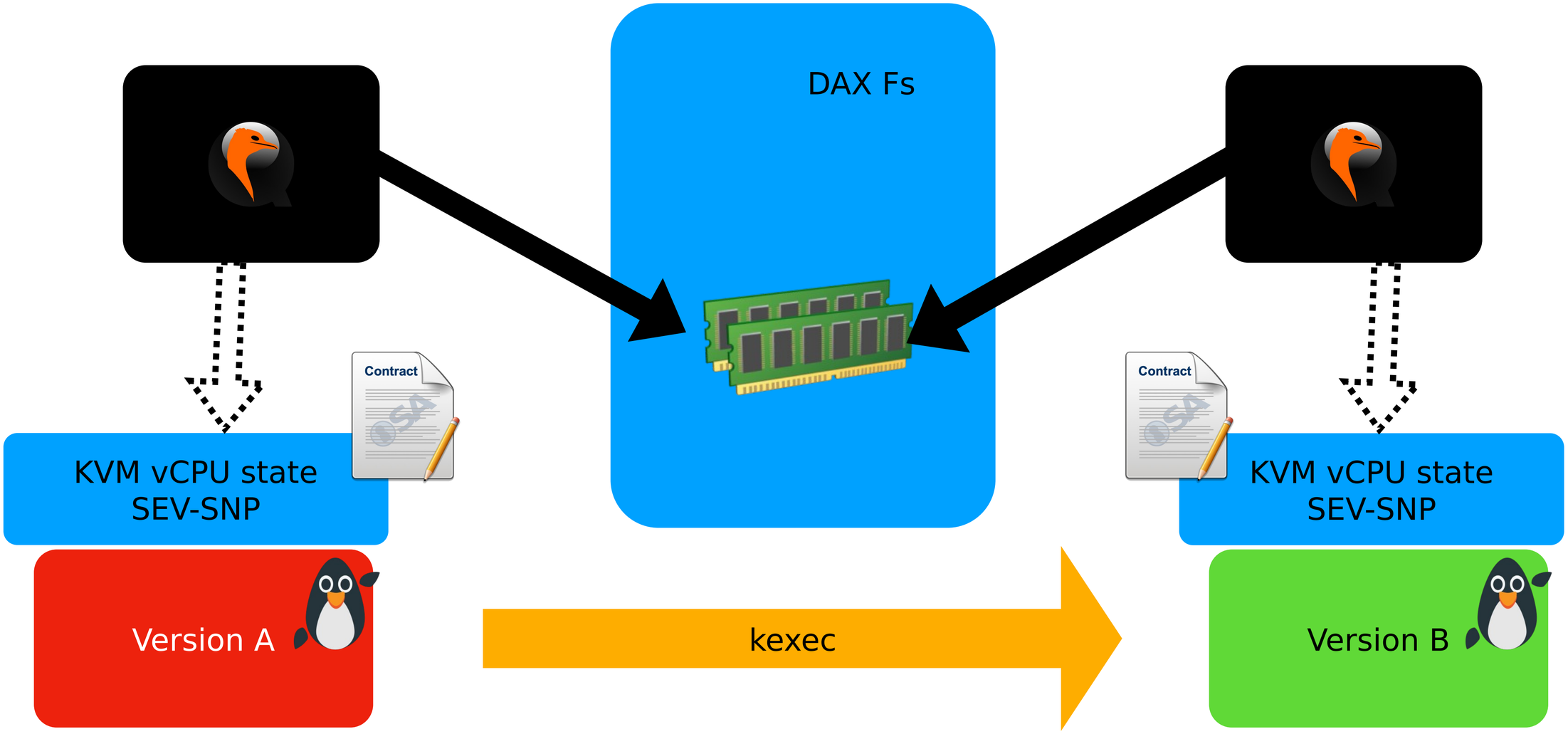


- IOMMU mappings
- User space handle for "old" mappings
- BAR mappings for P2P DMA
    - Prevent destructive operations

SEV-SNP: can’t serialise!



SEV-SNP: pass reference across



… others too.

- Combination of “correctness” and performance.
- PCI cache
- KVM “replay” state
- Calibration data

Proposals

Three “classes”:

- Memory pool: alloc/free
  - Reserved memory or dynamic memory

- Filesystem
  - Reserved memory or dynamic memory

- Serialise/deserialise framework supporting memory carve out

Memory pool

Two RFCs from Microsoft:

Persistent memory pool

- Reserve memory via cmdline param
- Alloc/free
- Not clear how to restore after kexec?
- https://lore.kernel.org/all/169645773092.11424.7258549771090599226.stgit@skinsburskii./

PRMEM

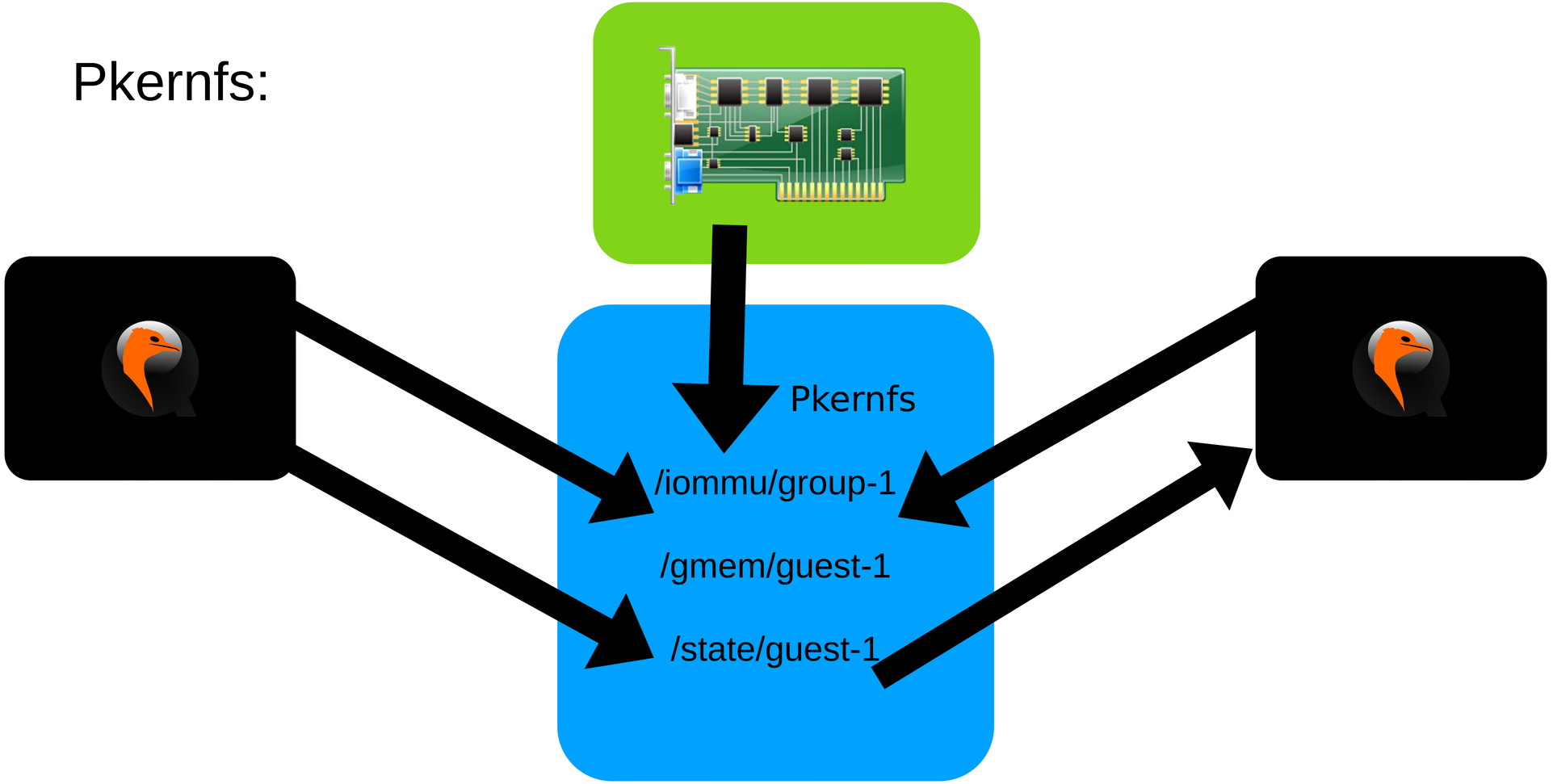
- Dynamic growable (not reserved)
- Pass pointer across kexec
- Supports key/value store
- https://lore.kernel.org/all/20231016233215.13090-1-madvenka@linux.microsoft.com/

Filesystem

- Filesystem on top of reserved memory: pkernfs
  - Proposal from AWS (RFC coming soon…)
  - Drivers able to open files and use data
  - Accessible to userspace too

- Filesystem on top of dynamic memory
  - PKRAM RFC by Oracle
  - Userspace only!
- https://lore.kernel.org/kexec/1682554137-13938-1-git-send-email-anthony.yznaga@oracle.com/

Pkernfs:


Pkernfs
/iommu/group-1
/gmem/guest-1
/state/guest-1


Version A
kexec
Version B

Serialise/deserialse

Kexec Handover

- Framework for drivers to hook into
- Describe state
- Preserved arbitrary memory pages.
  - Complexity: carve out of allocator; fragmentation
- Additional blob pointed to by setup_data (x86) or DT (arm)
- Proposal from AWS (RFC coming soon…)
- Simialar to Xen breadcrumbs.
  - http://david.woodhou.se/live-update-handover.pdf

Key differences

AKA: What should we do?

- Hard separation of “persisted” vs “ephemeral” memory?

- Solve all persistence with one solution or multiple solutions?
  - Specifically: kernel only or kernel and userspace

- Callers needs to alloc_persistent()?
  - Serialise: no; pool/fs alloc: yes.

Discussion

What should the userspace handles look like?

- IOMMU, KVM state, etc etc

Do we pass any memory or only reserved pools?

Options on metadata formats

Do we build an FS or metadata as base foundation?

More Discussion

Straying more into guest memory...

Userfaultfd on file system

Carve out chunks of VM memory and create a new VM out of them (Nitro Enclaves)

Direct Map hiding

Gmem for the file system -> No user space mapping