



RAM is getting more complex

Dr. David Alan Gilbert / dgilbert@redhat.com
Principal Software Engineer, Red Hat
2018-10-26

Topics

Types of memory

Command line memory specification

‘All of RAM/memory’

Sharing

Persistent memory

Types of host memory

QEMU data structures

Huge Pages

Encryption

Impact on migration

Types of memory

- RAM
- ROM
- Device memory
 - RAM in devices (e.g. Video RAM)
 - May have different alignment or caching rules
 - Emulated devices
 - Real devices
- Flash devices
 - Mostly like ROM, but with special indirect-write tricks
- Persistent memory
 - Mostly like RAM, but.....

Types of host memory

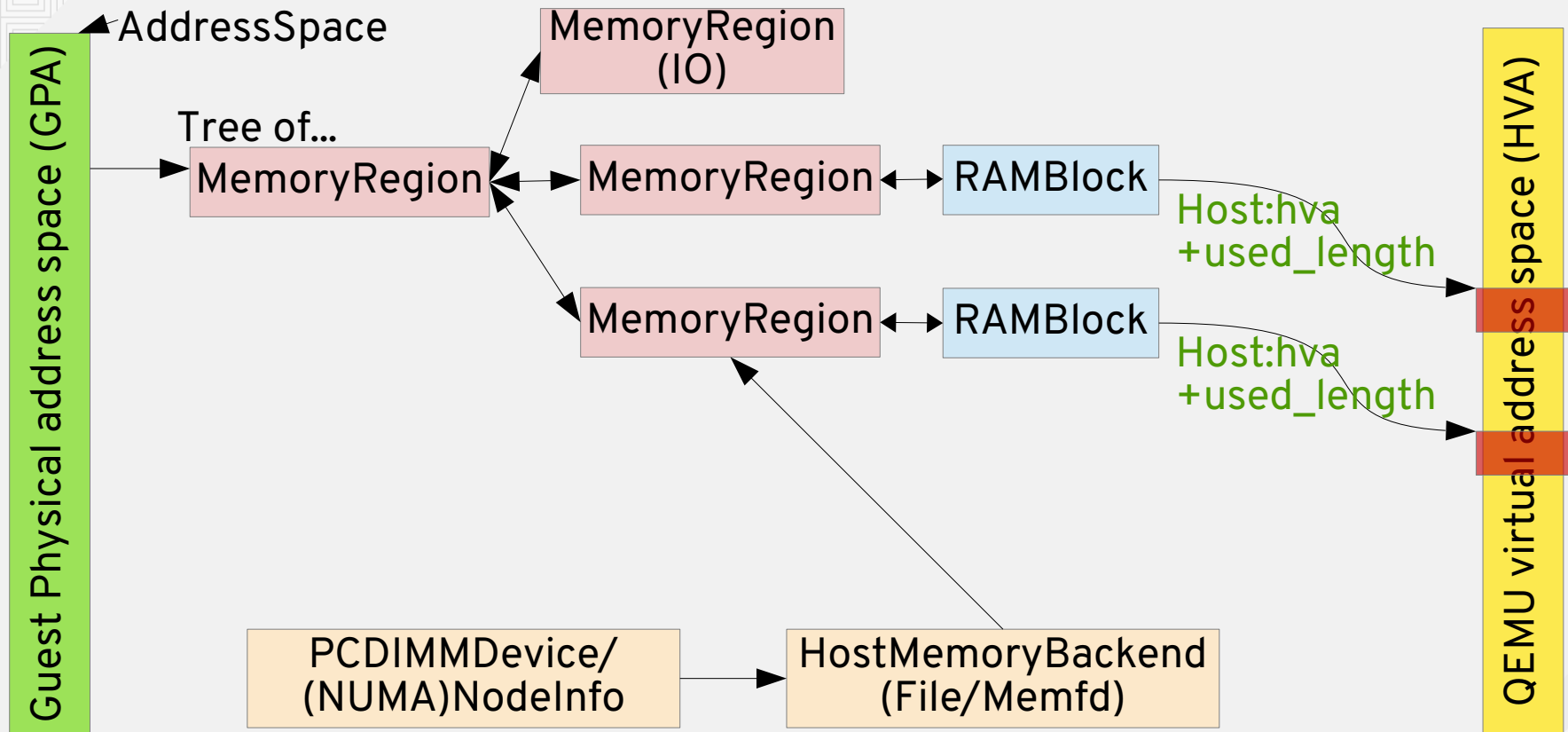
- Anonymous
 - Normal mmap
- File backed
 - 'file' hides many things:
 - Real files (rare)
 - Shmfs RAM filesystems
 - Hugetlbfs HUGE TLB pages
 - Persistent memory (pmem) backed
 - Note: ROMs are normally anonymous loaded from file

Command lines and memory

- `-m 4G` or `-m size=4G`
- `-m 4G, slots=3, maxmem=1T`
- `-mem-path /dev/hugepages`
- `-mem-path /dev/hugepages/foo`
- `-m 8G -object memory-backend-file, id=mem, mem-path=/dev/hugepages, size=8G -numa node, nodeid=0, memdev=mem`
- `-m 4G, slots=4, maxmem=16G -object memory-backend-ram, id=m1, size=1G -device pc-dimm, id=d1, memdev=m1`
- `-M ..., nvdimmm -m 4G, slots=4, maxmem=16G -object memory-backend-file, id=n1, size=4G, mem-path=/.../... -device nvdimmm, id=ndimmm1, memdev=n1`

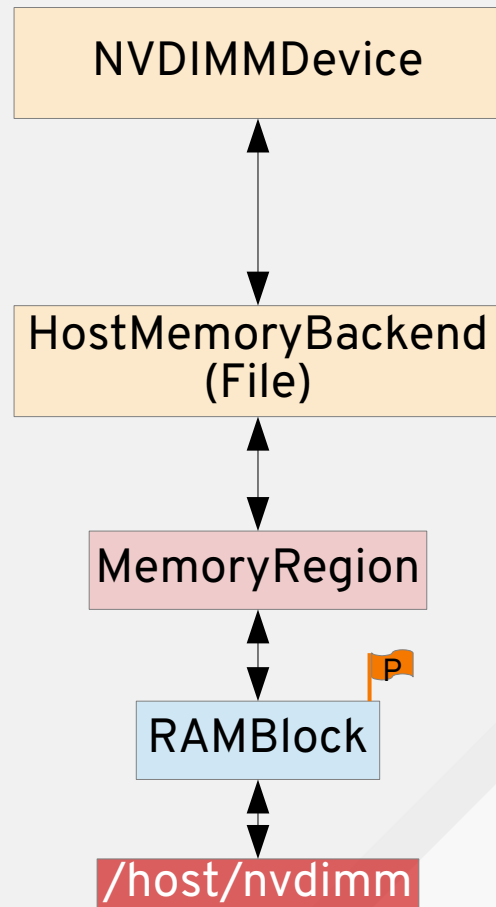
Falls back to normal memory unless
Used with `-mem-prealloc`

QEMU data structures



Persistent Memory (aka pmem)

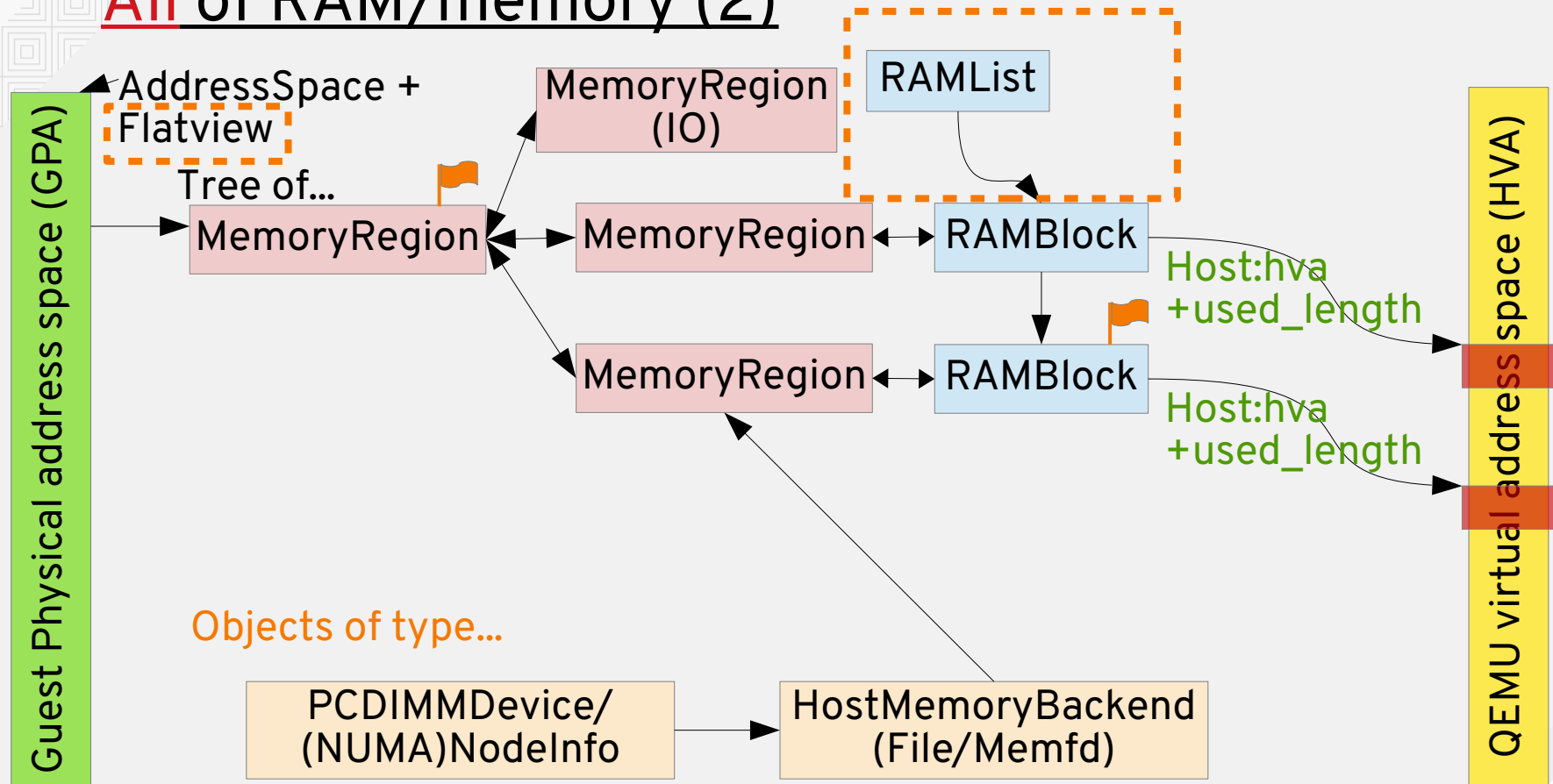
- Various rules to ensure consistency when accessing
 - See libpmem
 - QEMU must call libpmem after some of its own writes (e.g. migration)
- Typically come as 'NVDIMMs'
- Guest sees areas marked by ACPI
 - a) QEMU can use pmem as backing storage for RAMBlock's
 - b) QEMU can create virtual NVDIMMs in the guest
 - c) Can pass persistence flags to guest
- (a) & (b) are mostly independent
 - e.g. just use NVDIMM as more guest RAM and ignore persistence
 - e.g. fake NVDIMM as seen by guest



‘All of RAM/memory’

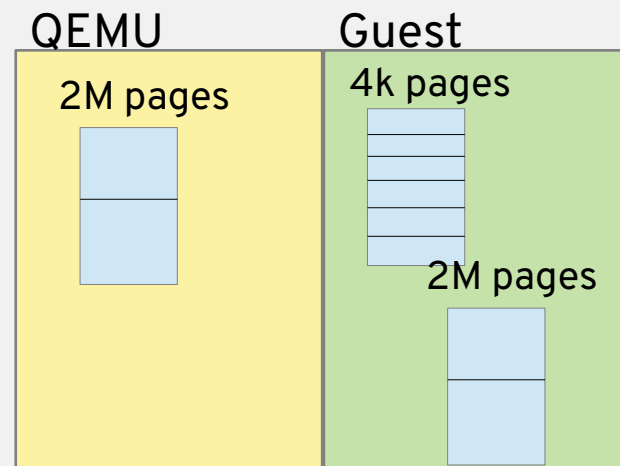
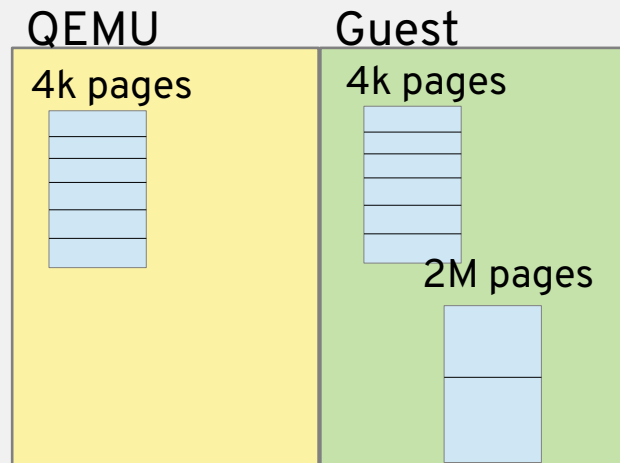
- All RAMBlock's?
- All guest visible ?
 - Not all RAMBlocks are mapped, some dynamically
- All of one address space?
- Include persistent memory?
 - Decided by backing or guest view?
- ROMs?
 - What about pflash?
- Video RAM?

All of RAM/memory (2)



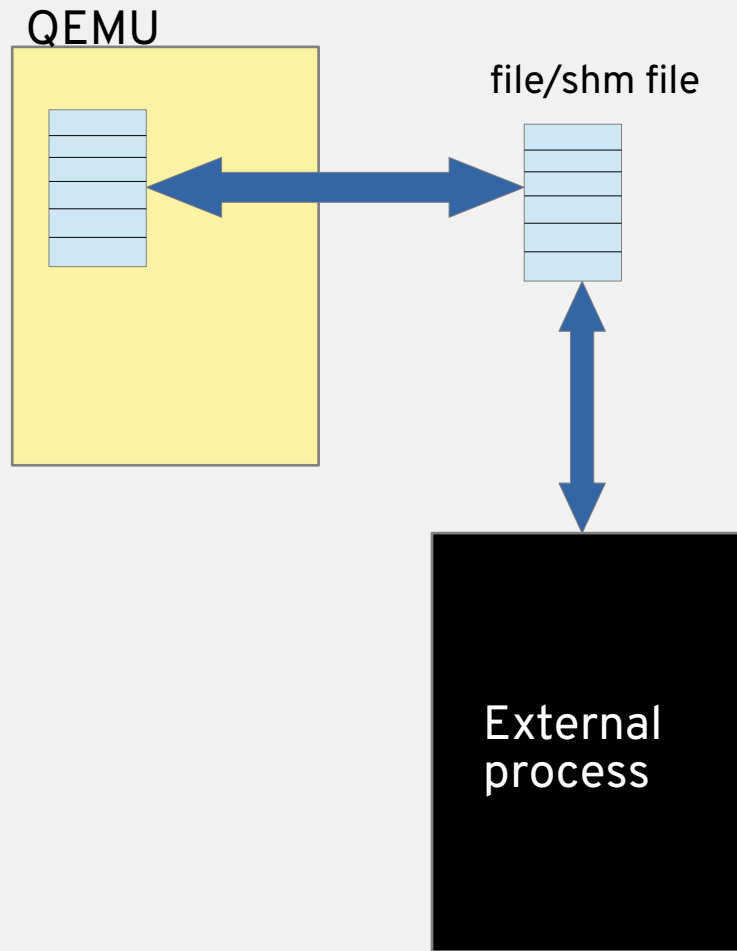
Huge pages

- Properties of individual RAM blocks
 - Can have a mix of some huge, some normal, different sizes of huge
- Guest and Host huge page are separate issues
 - Although some architecture specific restrictions (e.g. Power doesn't allow guest pages larger than host)
 - X86 allows any mix
- Page sizes architecturally dependent



Sharing

- Shared with another process
- Typically vhost-user
- Per-RAM block – but typically used for all main RAM when used.
- Difficult for QEMU to track users
- Sometimes shared by filename, sometimes by fd passing
- Needs some help for QEMU to track external dirtying



Encrypted guest RAM

- AMD's SEV
 - Most of guest memory encrypted
 - Not (usefully) readable by QEMU
 - Guest allows some areas to be accessed for IO
 - Process of measuring BIOSs etc

Migration

- Dirty page flags
 - At ‘target page’ granularity (typically 4k)
 - Some architectures dirty whole hugepage for one write
- Naming
 - RAMBlock names are part of the stream
 - Assigned only when marked for migration, typically when connecting **frontend** – can have unnamed RAMBlocks
- Postcopy needs kernel support for different backends
 - Now has normal, hugepage and shared support
 - Other backing files may need kernel support (e.g. pmem)

Useful HMP commands

- info ramblock

Block Name	PSize	Offset	Used	Total
pc.ram	4 KiB	0x000000000c800000	0x0000000040000000	0x0000000040000000
/objects/m1	4 KiB	0x0000000000000000	0x0000000006400000	0x0000000006400000
0000:00:02.0/vga.vram	4 KiB	0x000000004c880000	0x0000000001000000	0x0000000001000000

- info mtree

```
address-space: memory
0000000000000000-ffffffffffffffff (prio 0, i/o): system
0000000000000000-000000003fffffffff (prio 0, i/o): alias ram-below-4g @pc.ram
0000000000000000-000000003fffffffff
0000000000000000-ffffffffffffffff (prio -1, i/o): pci
00000000000a0000-00000000000bffff (prio 1, i/o): vga-lowmem
00000000000c0000-00000000000dffff (prio 1, rom): pc.rom
```

- info memdev
 - **Backends** e.g. HostMemoryBackendFile
- info memory-devices
 - **Frontends** e.g. PCDimm

Conclusion

- There are now lots of special cases
- Special types of host memory mapping
- Different types of memory devices visible to the guest
- Limitations on different architectures
- Different expectations on the lifetime/preservation of memory contents
- They can all combine into more special cases



THANK YOU