SNP Live Migration with guest-memfd & mirror VM

Pankaj Gupta

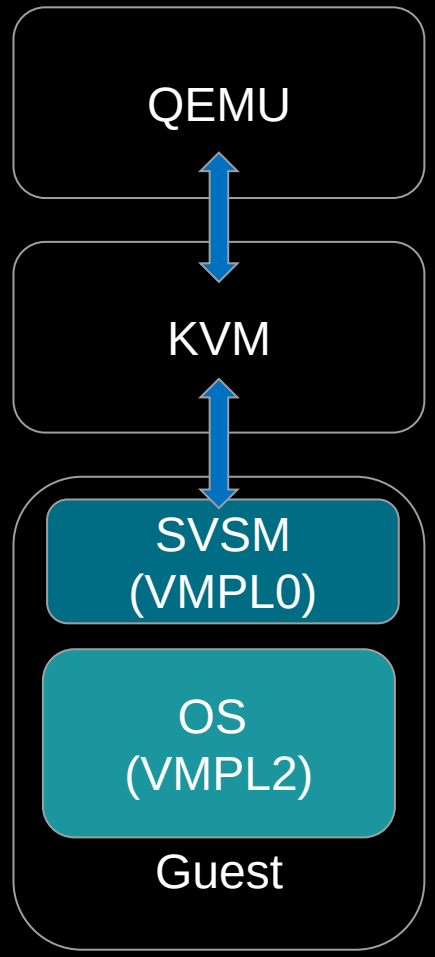
AMD
together we advance_

Agenda

- Introduction
- Approaches for SNP Live migration
  - Mirror VM
  - Shadow vCPU
- Open challenges
- Common vendors abstractions
- SNP live migration work

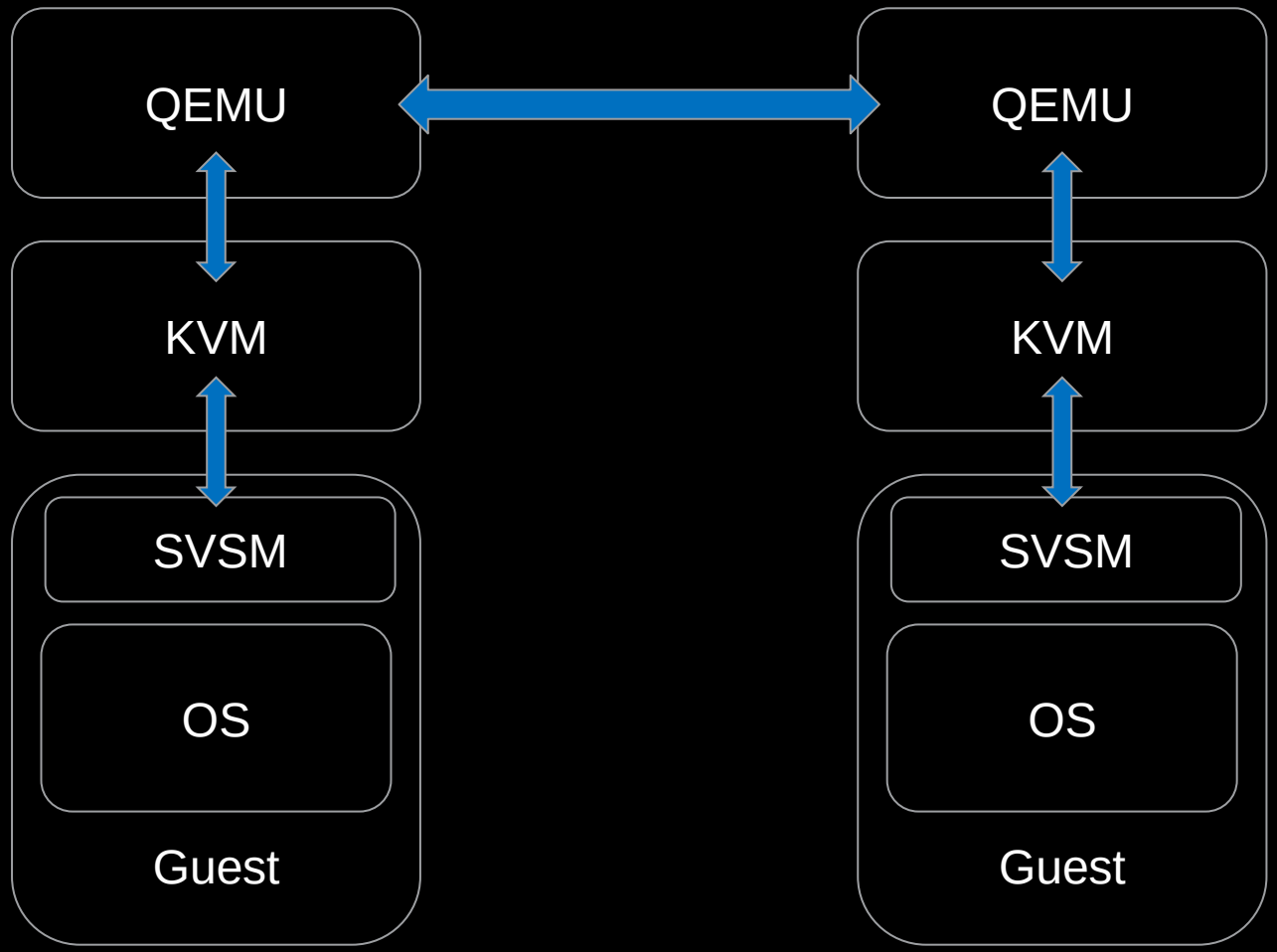
Introduction

- AMD SEV-SNP
  - Provides VM isolation & integrity protection
- VMPL (VM Privilege Level)
  - Each vCPU can run at different VMPL levels and hence have a corresponding VMSA (VM Save Area) per VMPL
  - Processor restricts guest memory access based on VMPL permissions
- SVSM (Secure VM Service Module)



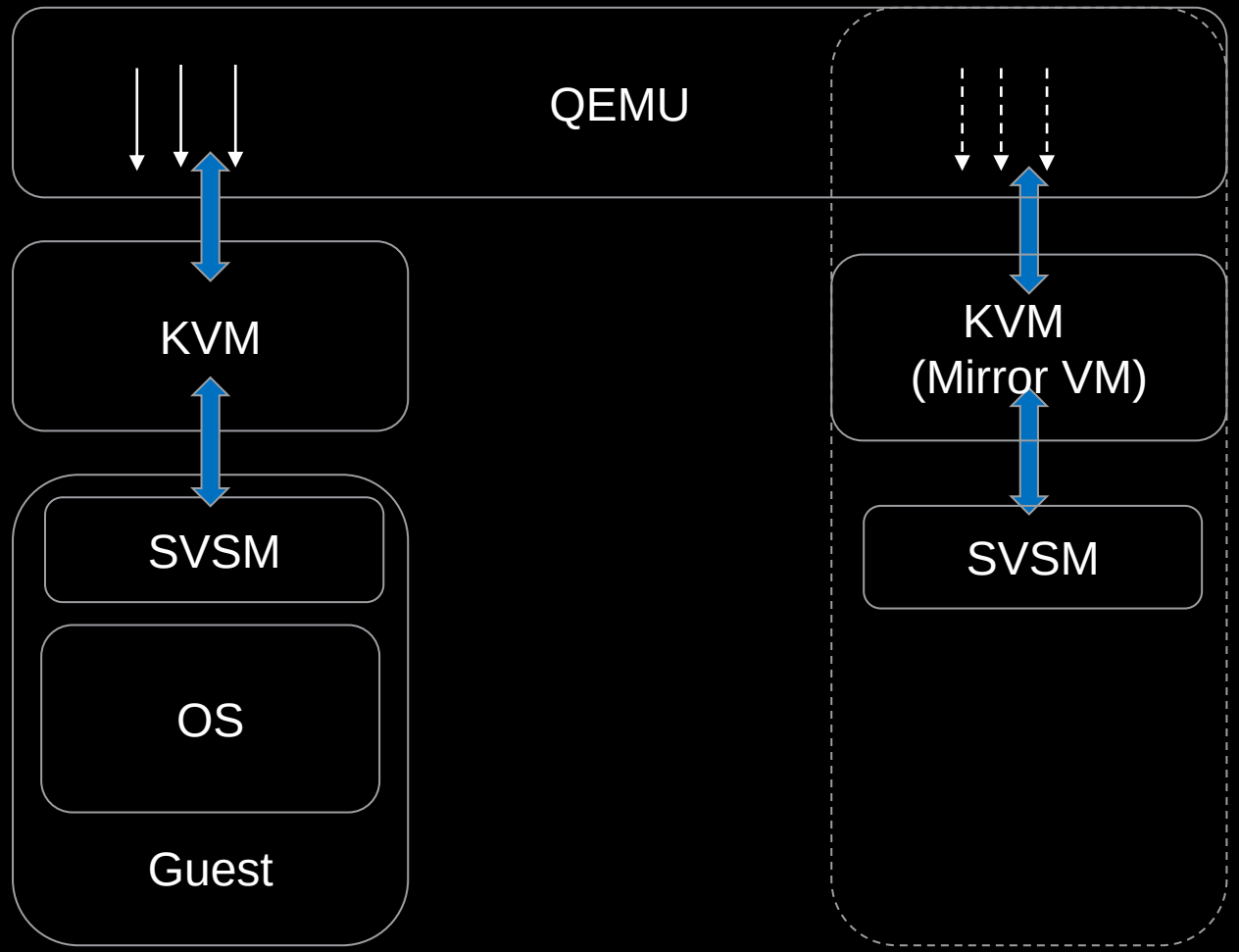
SNP Live migration

- SNP Live migration involves migrating running SNP guest from one host system to another with SNP enforced security guaranties.
- Guest private memory can only be read from the guest context.
- Need additional vCPUs; running at SVSM layer (VMPL0) and within the same guest context for live migration memory packaging.



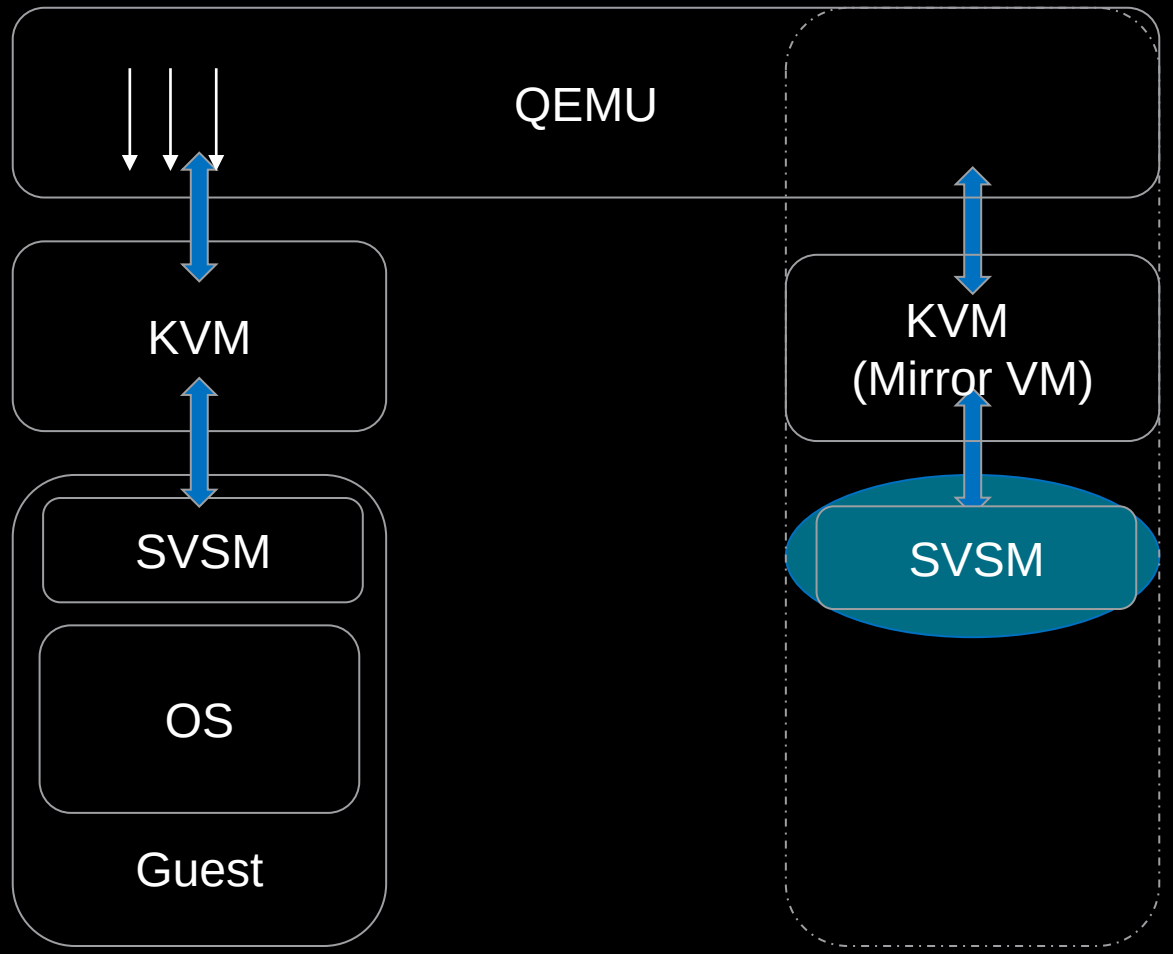
SNP Live migration approaches

- Mirror VM
- Shadow vCPUs



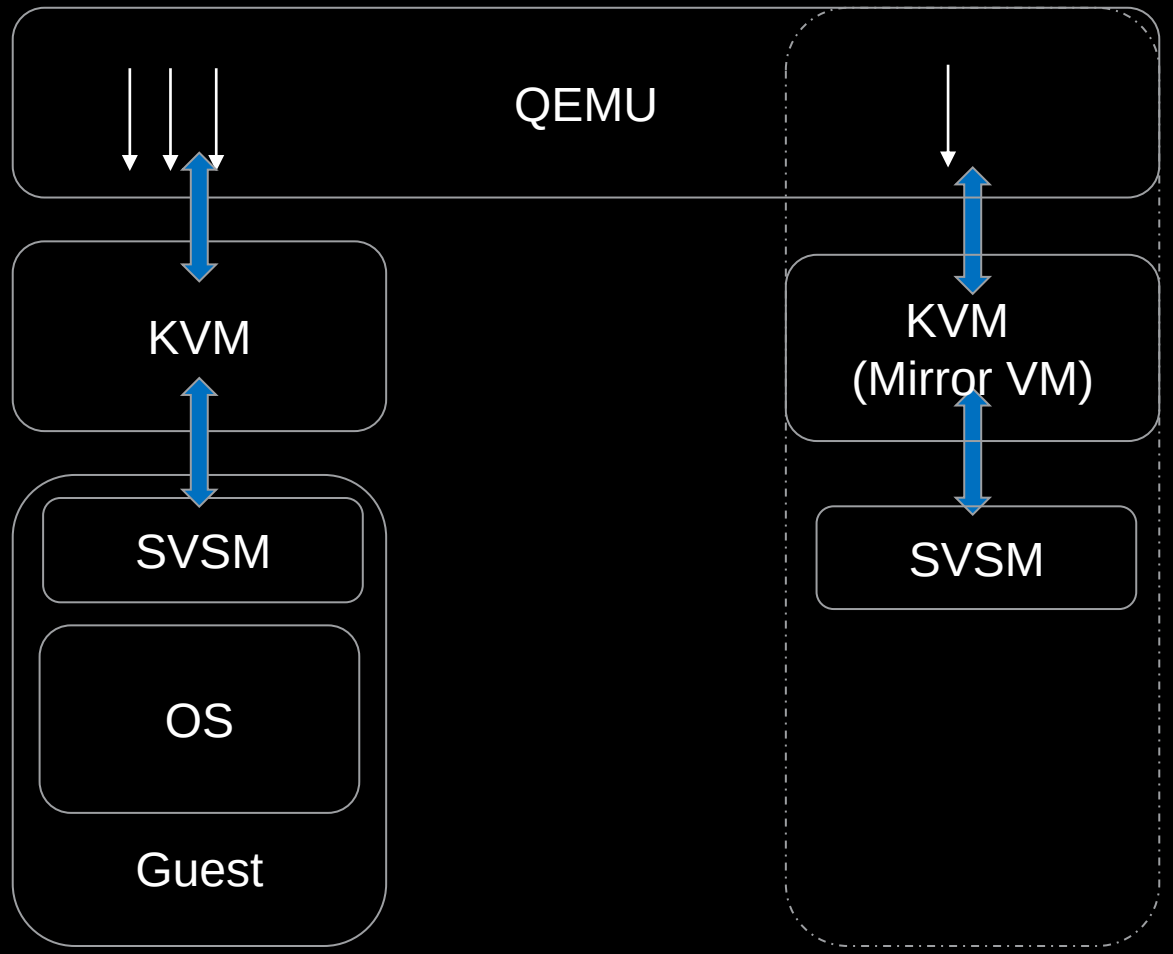
Mirror VM

- Mirror VM to perform the live migration for SEV-SNP guests.
- SVSM (Secure VM Service Module) runs at VMPL0 in the Mirror VM as migration helper & does the actual live migration related tasks.
- Host does the dirty bit tracking & SVSM does the memory packaging.



Mirror VM

- Mirror VM to perform the live migration for SEV-SNP guests.
- SVSM (Secure VM Service Module) runs at VMPL0 in the Mirror VM as migration helper & does the actual live migration related tasks.
- Host does the dirty bit tracking & SVSM does the memory packaging.
- First mirror vCPU is created dynamically at the live migration start time. Subsequent mirror vCPUs can be added on demand to scale the live migration.
- Qemu migration threads run as Mirror vCPUs and are dedicated only for live migration tasks.

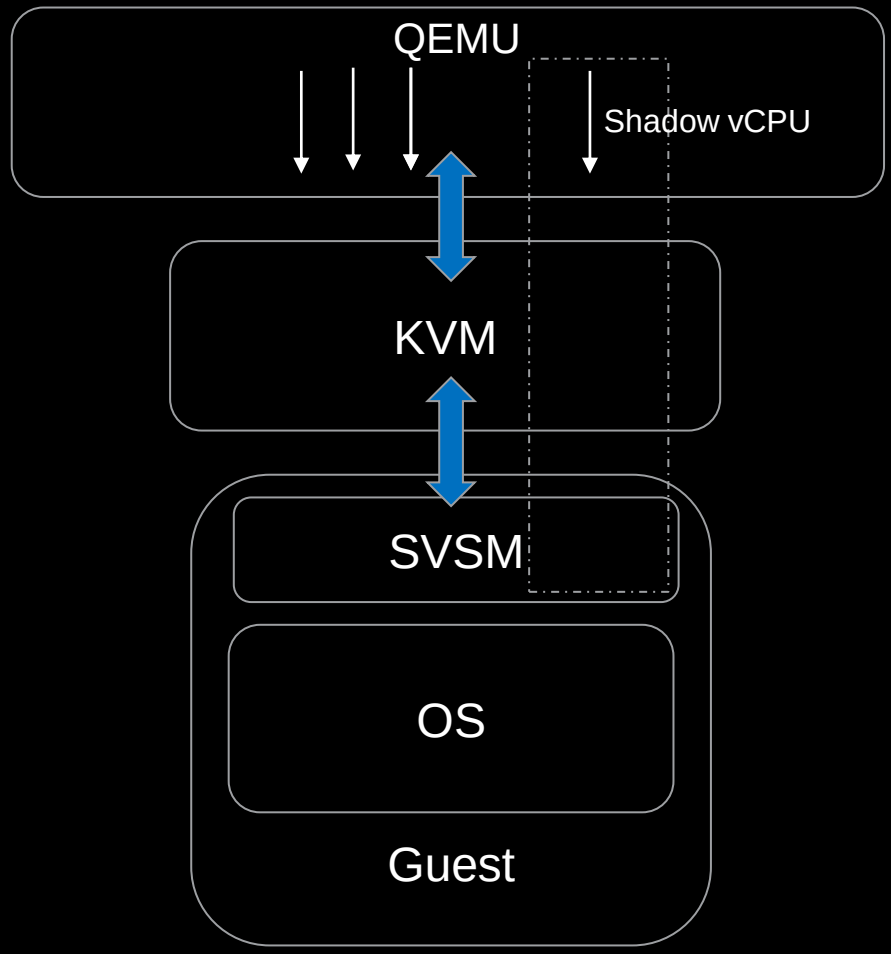


Current State of Mirror VM

- Guest-memfd does not support address space sharing between mirror & primary VM.
  - Guest-memfd in current state does not handle multiple KVM objects, as guest memfd is tightly bound to single KVM object.
  - Post-copy migration also needs to be compatible with guest-memfd address sharing.
- Needs something like VMA like address space sharing between primary & mirror VM for the guest-memfd interface.
- [RFC PATCH 00/11] New KVM ioctl to link a gmem inode to a new gmem file - Ackerley Tng
  https://lore.kernel.org/lkml/cover.1691446946.git.ackerleytng@google.com/
- Needs input from the community.

Alternative approach: Shadow vCPUs

- Shadow vCPUs need non-overlapping APIC_ID with the primary VM to create the APs.
- AP creation into VMPL0; involves dynamic VMSA creation.
- Primary VM does the migration using the shadow vCPUs of the primary VM itself.



Unifying Coco VM Live Migration

- Vendor Neutral uAPIs/KVM ioctls for live migration functions
- Attesting the target
- Integrity protection during live migration
- Guest_memfd
  - New APIs for memory pre-copy live migration
  - Post-copy live migration support

SNP specific tasks

- Initial handshake with the help of attestation to validate the participating guests and exchange the migration keys
- Memory packing & Dirty bit logging.
- VMSA and other VM state transfer
- Integrity protection during live migration and end of live migration validation

Summary

- Mirror VM
  - Separates the responsibility to do the SNP live migration to a transient VM
  - Better accounting for Cloud vendors
  - By default, separate KVM object for vCPUs in Mirror VM - [TBD]
    - guest_memfd needs support for multiple KVM objects
    - KVM level address space sharing support

- Shadow vCPU
  - Qemu migration thread run as shadow vCPUs, using host CPUID (Common Mirror VM) – [Code done]
  - Non-overlapping APIC_ID range support in KVM for shadow AP creation – [TBD]
    - Create APs in VMPL0 from host (Common with Mirror VM)
  - No separate KVM object for shadow vCPUs

References

- SEV-SNP Live Migration and VMM/KVM API Implications
- Secure Live Migration of Encrypted VM
- Mirror VM POC for SEV
- KVM: Post-copy live migration for guest_memfd
- Guest memfd discussion on guest-memfd based shared memory & hugepages

DISCLAIMER

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED 'AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AMD, the AMD Arrow logo, AMD-V and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.


AMD