Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024

Intel TD Partitioning and vTPM on COCONUT-SVSM

Peter Fang, Intel

Chuanxiao Dong, Intel

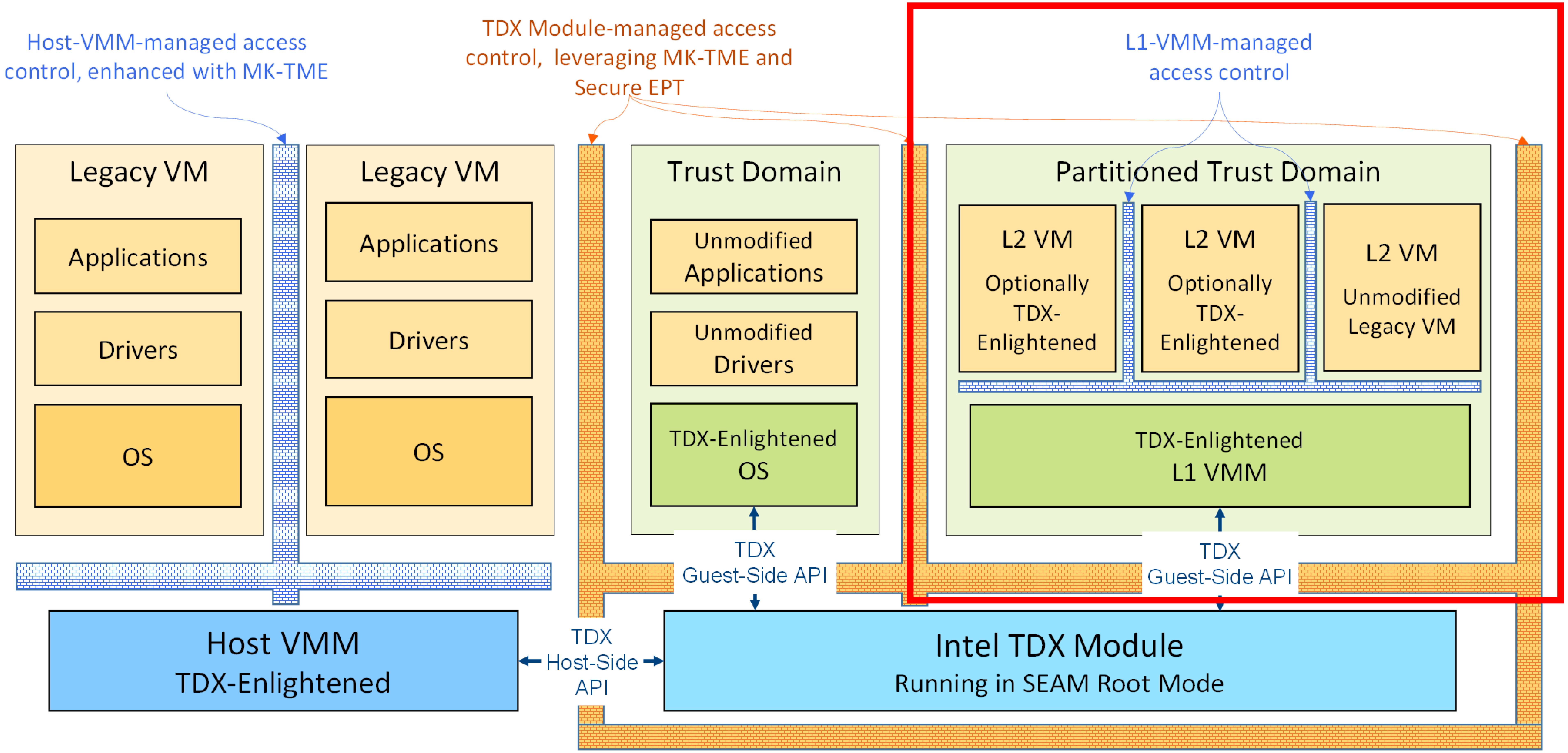
Jiewen Yao, Intel

LINUX PLUMBERS CONFERENCE Vienna, Austria / Sept. 18-20, 2024

Agenda

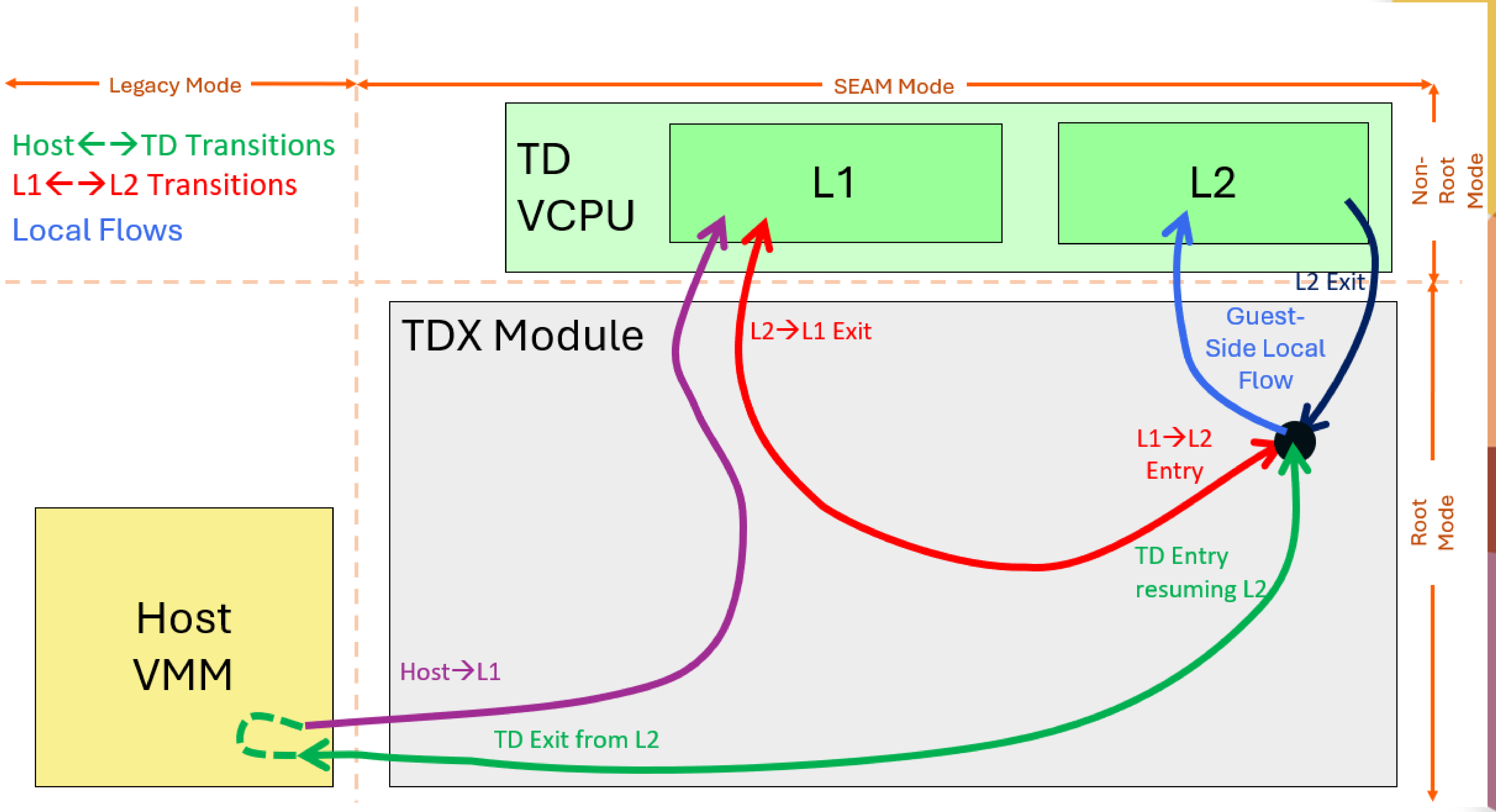
- Overview of Intel TD Partitioning
- COCONUT-SVSM TDP status update
- TDP-based vTPM

Architecture Overview



L2/L1 vCPU Transitions

- A TD vCPU is running in either L1 or L2 at any given time
- Host VMM can initiate host→L1 or host→L2 entry
- L1 can initiate L1→L2 entry
- L2 VM exit
  - Causes TD exit (handled by host VMM)
  - Causes L2→L1 exit (handled by L1 VMM)
  - Guest-side local flow (handled by the TDX module alone)



L2 Interrupt Virtualization

- x2APIC mode is virtualized via APICv (a virtual APIC page is available)
- xAPIC mode is emulated by L1 VMM via software
- Posted interrupts for L2 VMs are not supported

L2 Private Memory Virtualization

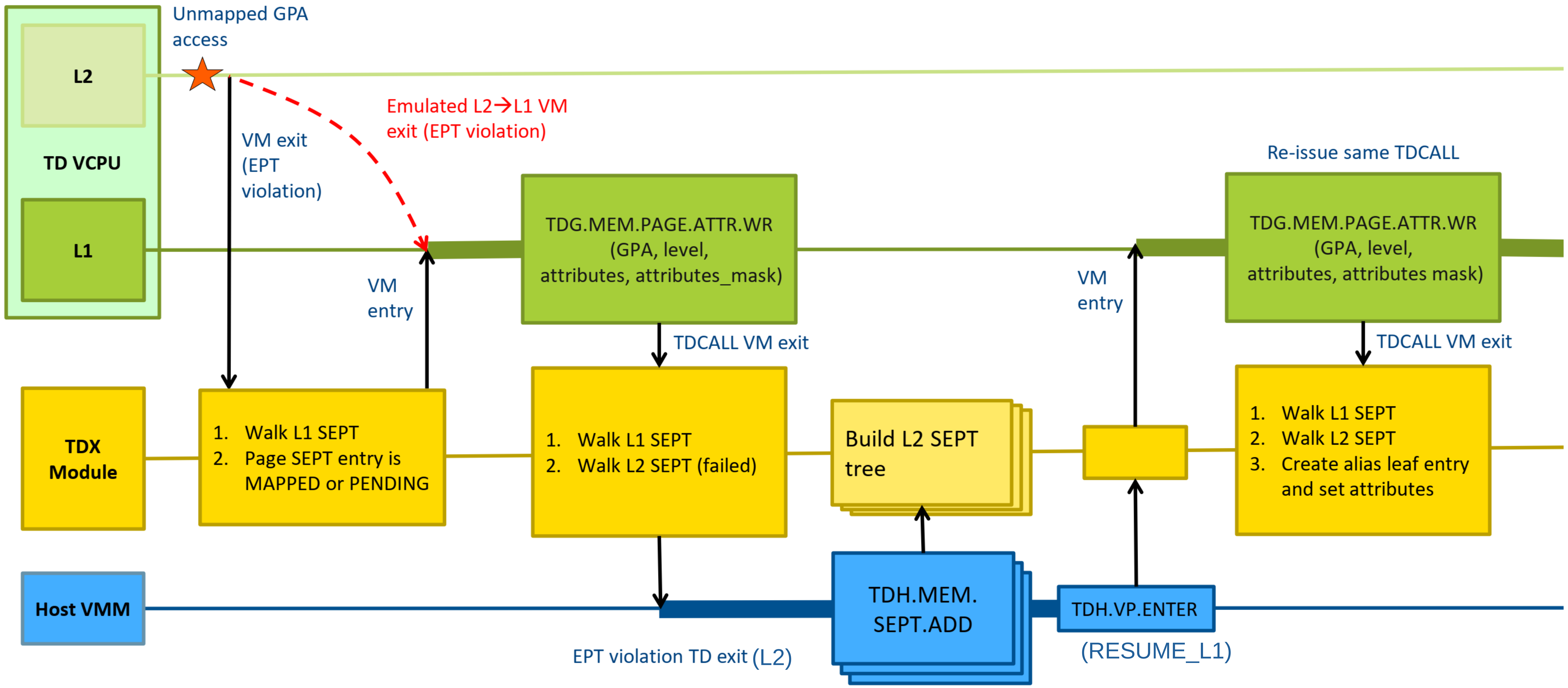
- To simplify L2 memory management, TDP L2s use *page aliasing* to partition the GPA space (no separate L2 GPA→L1 GPA mappings)
- Each L2 SEPT is individually managed but L1 and all L2s share the same GPA space
- L1 VMM manages L2 page aliases through *TDG.MEM.PAGE.ATTR.WR*

* L2 shared memory is treated in the same way as L1 shared memory

GPA Space	**VM #0 L1 VMM**		**VM #1 L2 VM (e.g. Chrome)**		**VM #2 L2 VM (e.g. TEE VM)**		**VM #3 L2 VM**
Page A	RWX		R				None
Page B	RWX		RW		RW		R
Page C	RWX		RWX		None		
Page D	RWX		None		RWX		
Page E	RWX		RWX				R
Page F	RWX		RWX				R
Page G	RWX		RW				R
Page H	RWX		RWX				R

Putting It All Together…

- Example: Adding an L2 page alias



TD Partitioning (L2) vs TDX (L1)

- TDP Supports all CPU modes supported by VMX non-root mode (real mode, protected mode, compatibility mode, long mode).
- A TDP guest is much more similar to a traditional VMX guest; most x86 instructions can be executed in the guest.
- TDP requires less enlightenment. It’s possible to have a completely unmodified TDP guest, albeit there would be performance degradation. Comparable performance can be achieved by enlightening the guest to support shared pages and GHCI.
- TDP has L1 VMM in its TCB. Secure device models such as vTPM can exist in L1 VMM.

TD Partitioning vs Traditional Nested Virtualization

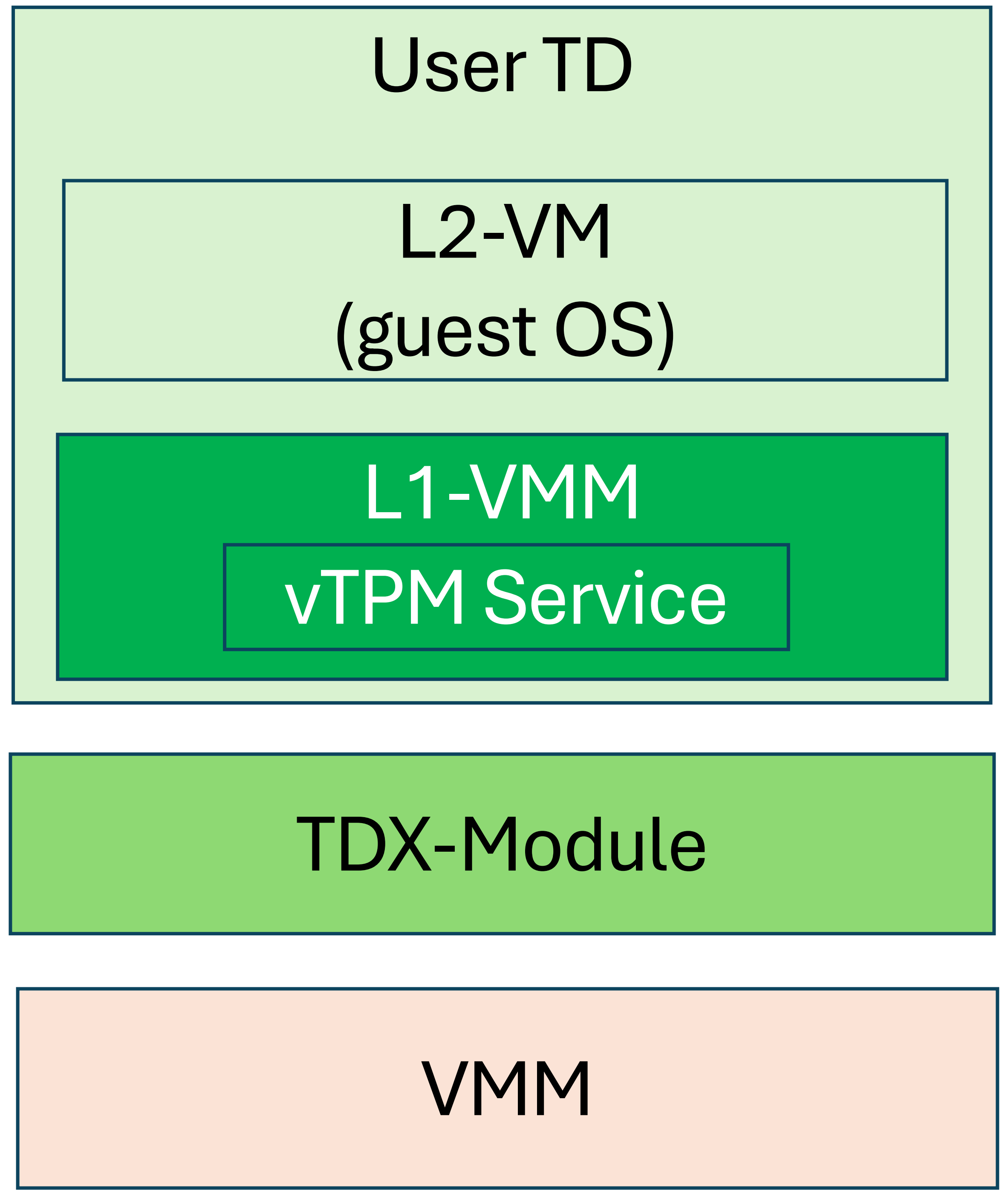
- TDP achieves security through two L0 hypervisors: host VMM (non-SEAM mode) and the TDX module (SEAM root mode).
  - TDP L2 exit flows are more complex: local flow + TD exits + L2→L1 exits (all are vmexit-esque)
- TDP simplifies guest memory management by adopting page aliasing.
- VMX instructions are disallowed in L1 TD; L1 VMM uses TDCALL instructions (TDG calls) instead.
  - TDG calls are mostly akin to VMX instructions but also include TDX-specific extensions.

COCONUT-SVSM TDP Status

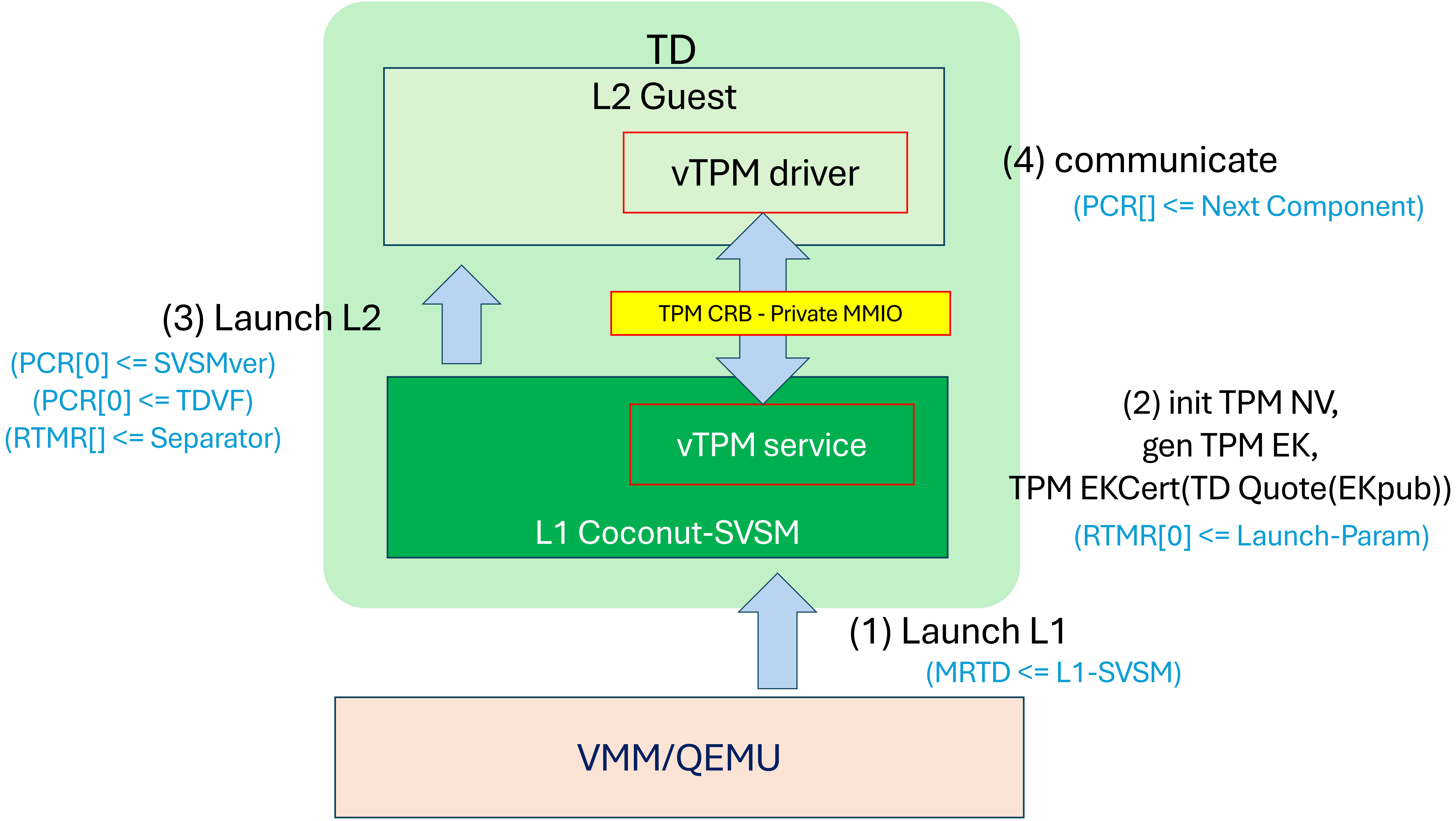
- Demo code published on GitHub (boots vanilla Linux kernel as L2)
- TDX enabling partially upstreamed (IGVM support, stage1, part of stage2; SVSM kernel and SMP support pending)
- Actively engaging with the community to provide vTPM, instruction decoder and user-mode support
- Highlights & challenges
  - Enabling: TDX boots into stage2 via IGVM now but more enabling is needed. Working with upstream stakeholders on platform abstraction to reduce TDX-specific logic.
  - Interrupt: Spec to inject interrupts from host to SVSM and from SVSM to L2 are mostly finalized. Need to engage with KVM maintainers to get their buy-in and upstream TDP restricted injection patches.
  - User mode: Uploaded drafts for user-mode VMM design and syscall object management framework. Working with the community to start the code review & upstreaming process.

Intel TD Partitioning based vTPM solution on Coconut-SVSM

TD-Partition Based vTPM



High Level Archtecture



Role in vTPM TD-Partitioning solution

Role	vTPM Service
Virtual Root of Trust for Reporting (vRTR)	**vTPM Service**: TPM software stack.
Virtual Root of Trust for Storage (vRTS)	**vTPM Service**: vTPM non-volatile storage (NVS) inside of coconut-SVSM. NVS is actually not persistent.
Virtual Root of Trust for Measurement (vRTM)	**L1 coconut-SVSM**: extend initial TDVF to PCR[0] (Similar to Intel Boot Guard ACM)
vTPM Endorsement Key (EK) Certificate	**vTPM Service**: generate key pair inside of NVS. **Self-signed EK Cert**: OID:“vTPM coconut SVSM Quote” in the certificate – hash of EKpub is included in the TdQuote.

Ephemeral vTPM only

- **No Persistent Storage in vTPM**

  By default, persistent storage disappeared after coconut-SVSM teardown.

- **vTPM NVS (Non-Volatile Storage)**

  Ephemeral NVS is implemented inside of vTPM service in coconut-SVSM.

- **vTPM EK**

  Ephemeral EK generated when SVSM init.

  EKpub hash is included as REPORTDATA in TDREPORT for coconut-SVSM.

MR/PCR life cycle

L2 Guest

L1 Coconut-SVSM

L1 TD Quote
MRTD=||**coconut-SVSM**
RTMR[0]=||**Launch-Param**
RTMR[1]
RTMR[2]
RTMR[3]=0
REPORTDATA=H(EK_{Pubkey})

vTPM PCR(T0)
PCR[0]=0
PCR[1~7]=0
PCR[8~16]=0
PCR[17~23]=0

T0 VMM launch L1 Coconut-SVSM

T1 L1 Coconut-SVSM launch L2 Guest

L1 TD Quote
MRTD=||coconut-SVSM
RTMR[0]=||LP||Separator
RTMR[1]=||Separator
RTMR[2]=||Separator
RTMR[3]=||Separator
REPORTDATA=H(EK_{Pubkey})

L2 Guest vTPM PCR(T1)
PCR[0]=||**SvsmVer||TDVF**
PCR[1~7]=0
PCR[8~16]=0
PCR[17~23]=0

T2 TDVF:TPM2_PCR_Extend(PCRIndex, tdvf_data)

T3 OS:TPM2_PCR_Extend(PCRIndex, os_data)

L2 Guest vTPM PCR(T2)
PCR[0]=||SvsmVer||TDVF
PCR[1~7]=||tdvf_data
PCR[8~16]=0
PCR[17~23]=0

L2 Guest vTPM PCR(T3)
PCR[0]=||SvsmVer||TDVF
PCR[1~7]=||tdvf_data
PCR[8~16]=||os_data
PCR[17~23]=0

Attestation Architecture

- vTPM EK cert contains the TD_Quote.

- TD_Quote reflects L1 info and provides authenticity of vTPM.

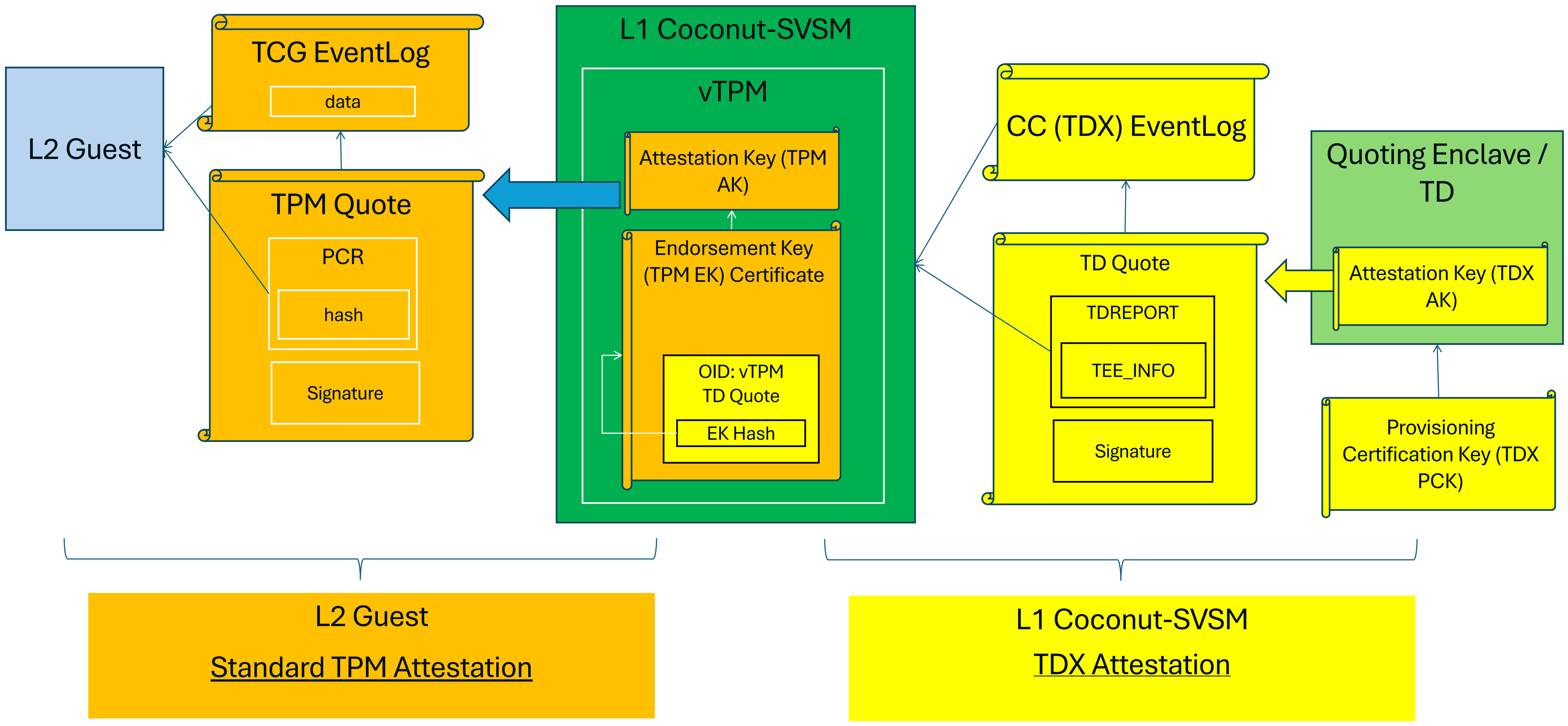
  MRTD/RTMR == L1 coconut-SVSM

  REPORTDATA == L1 vTPM EK.

- vTPM PCR reflects L2 TD measurement.

  L2 TDVF is measured into PCR0, by RTM (L1 coconut-SVSM).

Combined Attestation



References

- Intel TDX Module v1.5 TD Partitioning Architecture Specification: https://www.intel.com/content/www/us/en/content-details/773039/intel-tdx-module-v1-5-td-partitioning-architecture-specification.html
- Guest-Host-Communication Interface (GHCI): https://www.intel.com/content/www/us/en/content-details/726790/guest-host-communication-interface-ghci-for-intel-trust-domain-extensions-intel-tdx.html
- Intel TD-Partitioning based vTPM document: https://github.com/intel-staging/td-partitioning-svsm/blob/svsm-tdp-vtpm/Documentation/TD%20Partitioning%20based%20virtual%20TPM%20Design%20Guide%20Rev%200.5.1.pdf
- COCONUT-SVSM: https://github.com/coconut-svsm/svsm
- Intel's SVSM-TDP PoC: https://github.com/intel-staging/td-partitioning-svsm/tree/svsm-tdp
- Intel TD-Partitioning based vTPM POC: https://github.com/intel-staging/td-partitioning-svsm/tree/svsm-tdp-vtpm

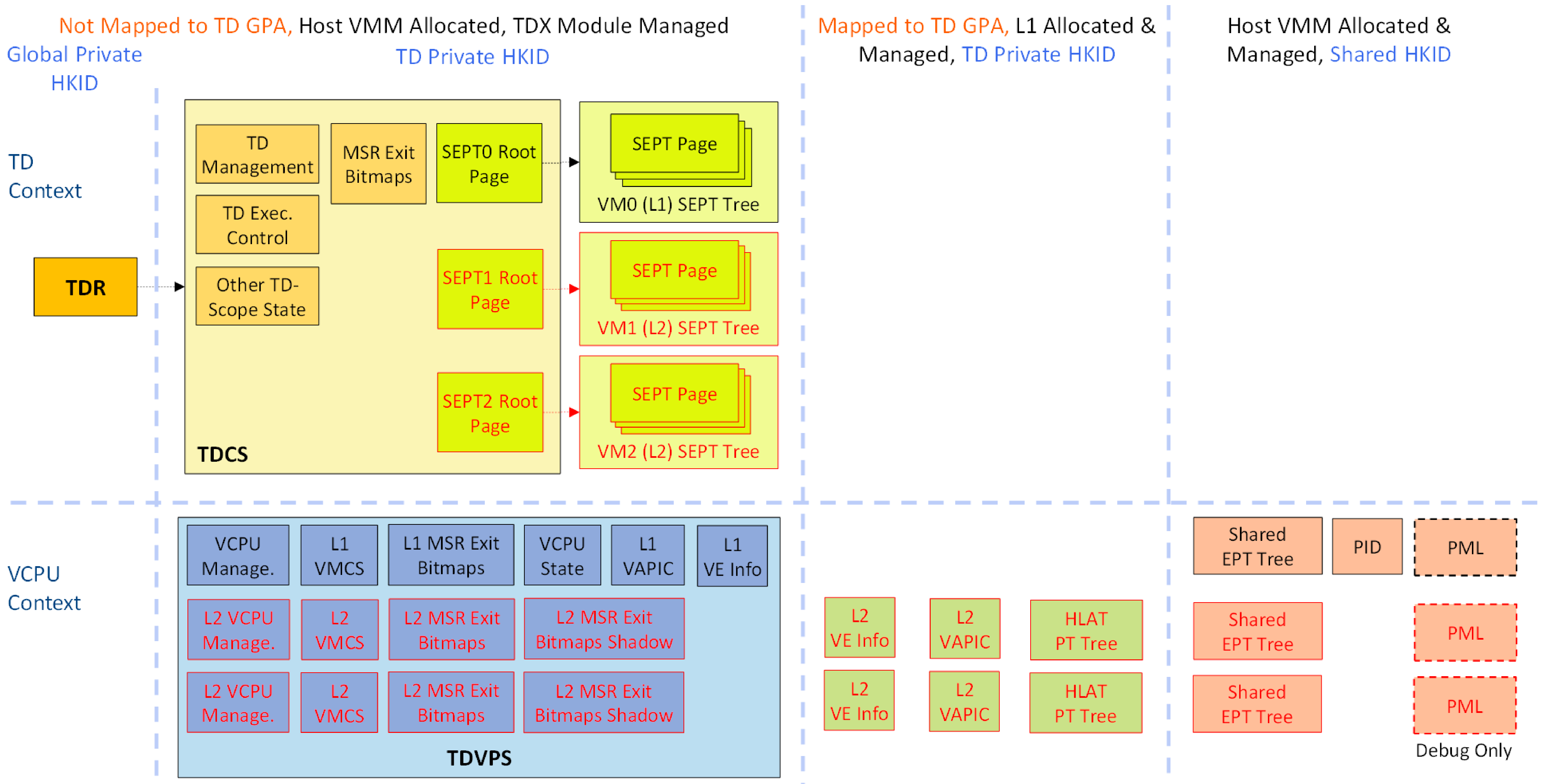
Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024

Backup Slides

Control Structures for L2 VM

- Host VMM allocated, TDX module managed
- Host VMM allocated, host VMM managed
- L1 VMM allocated & managed



L2 VM Exits

- L2 VM exits are always caught by the TDX module first
- L1 VMM handles most of the remaining exits via L2→L1 exits
- The TDX module handles the most critical cases (e.g. sensitive MSR/CR accesses, etc)
- A few are handled by host VMM (e.g. NMI, external interrupt, SEPT-related EPT violations, etc)