Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024

Mediated Passthrough vPMU for KVM

Presenter: Mingwei Zhang (Google)

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

Agenda

- Introduction to vPMU
- Motivation of Passthrough
- Design of Mediated Passthrough vPMU
- Current Status and Next Steps
- Q&A

Acknowledgement / co-authors

Google	Ian Rogers, Jim Mattson, Lotus Fenn, Raghavendra Rao Ananta, Sean Christopherson, Stephane Eranian
Intel	Dapeng Mi; Kan Liang; Samantha Alt; Xiong Zhang; Zhenyu Wang
AMD	Manali Shukla, Nikunj Dadhania, Sandipan Das, Santosh Shukla
Community	Bibo Mao, Namhyung Kim, Peter Zijlstra, Yongwei Ma, Zide Chen,

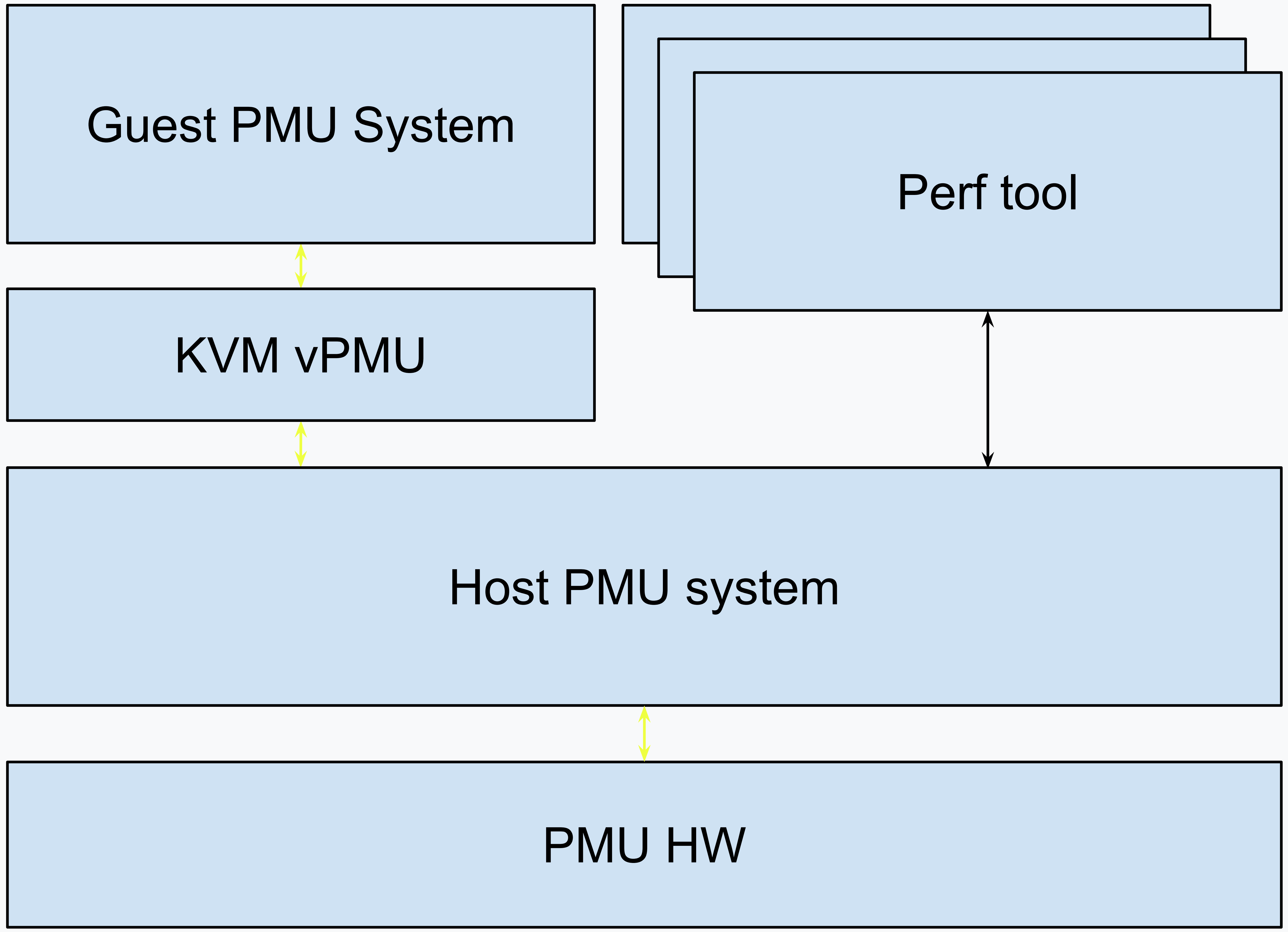
Existing vPMU in KVM

Existing KVM vPMU implementation is built on top of host PMU

- Guest VM accesses to PMU VMexits to KVM
- KVM wraps everything and send API request to host PMU
- Host PMU access HW PMU and get it back

Accessing PMU MSRs from guest is a long journey

KVM vPMU is a client to Host PMU system



Existing vPMU in KVM

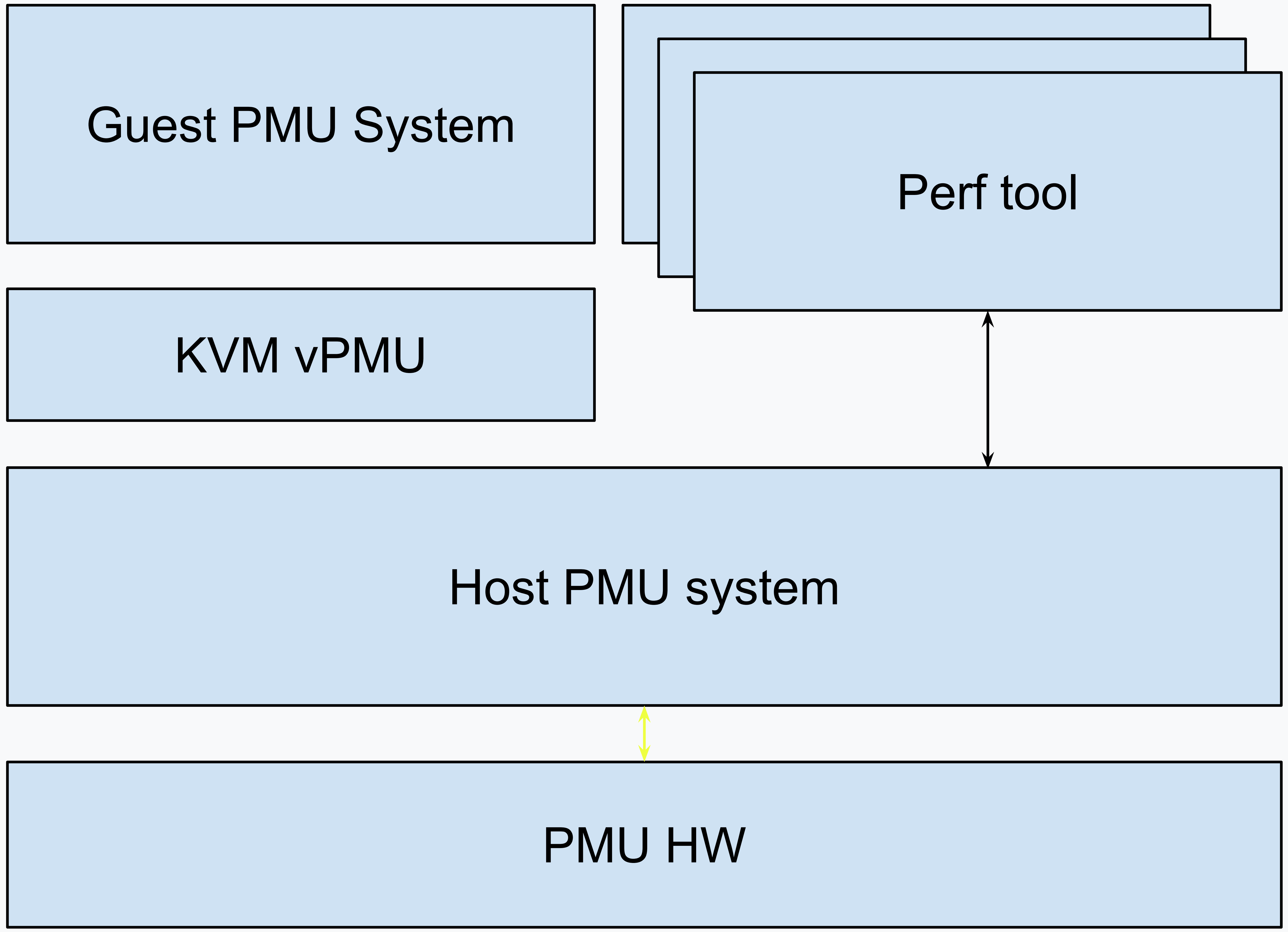
Existing vPMU implementation has the following issues

- **High performance overhead**
- Difficult to extend with advanced PMU features.
- High maintenance cost.

Guest PMU generates lots of PMU reset thinking it cheap as one single MSR write.

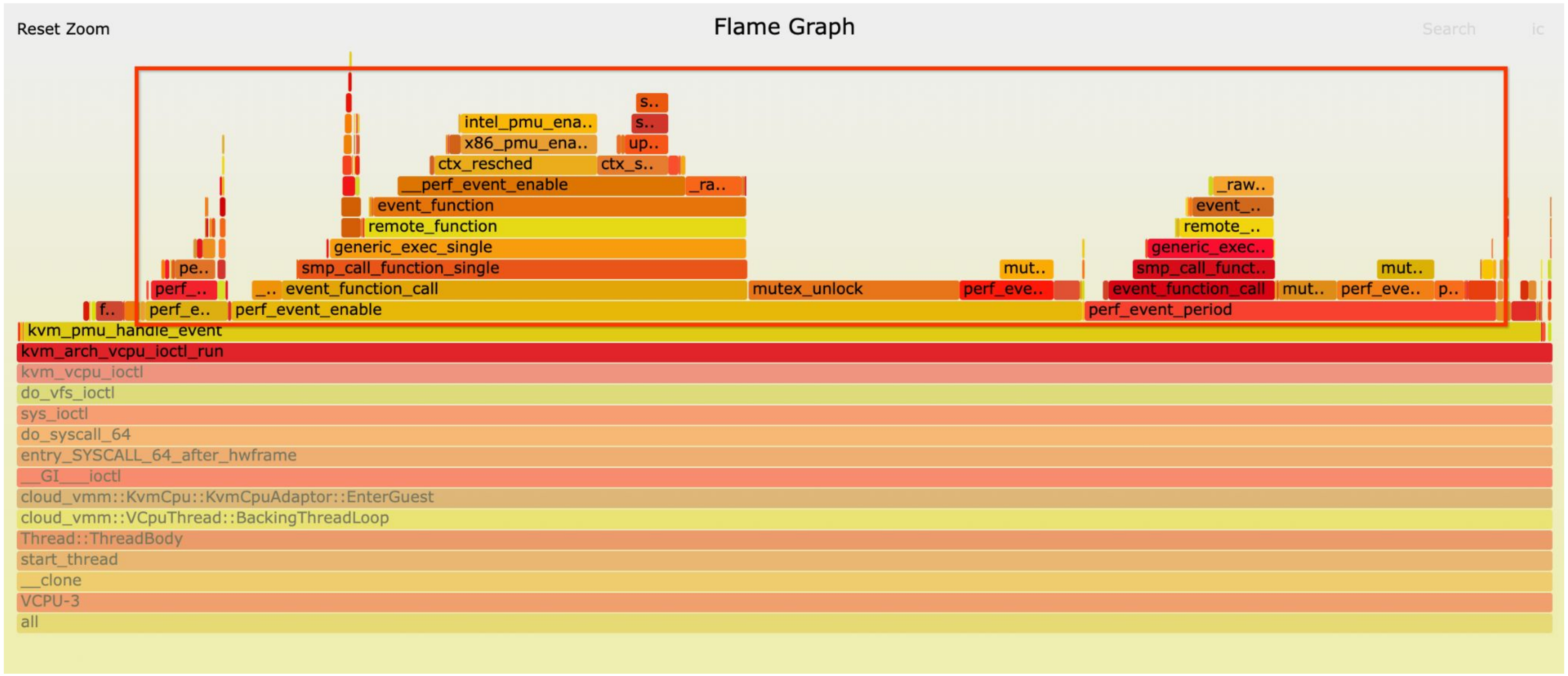
KVM vPMU wasted lots of time pausing and resuming counters

Host PMU system spend lots of time on internal routines.



Majority of the time is wasted in

- **perf_event_enable()**
- **perf_event_period()**
- **perf_event_create_kernel_counter()**
- find_next_bit()



Google

Existing vPMU in KVM

Existing vPMU implementation has the following issues

- High performance overhead
- **Difficult to extend with advanced PMU features.**
- High maintenance cost.

PEBS as an advanced PMU feature will "Intermittently" available across VMExit

Google

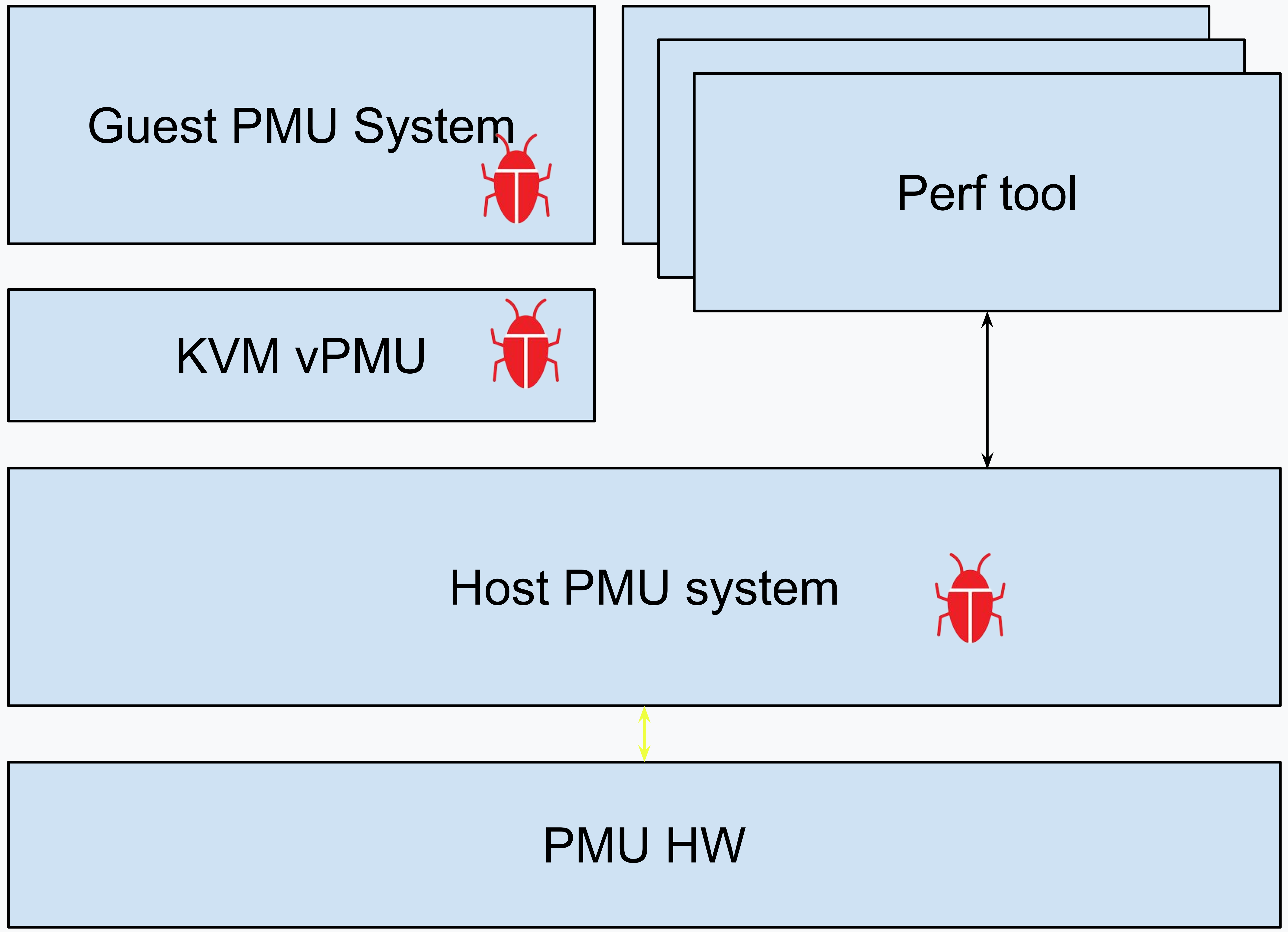
Existing vPMU in KVM

Existing vPMU implementation has the following issues

- High performance overhead
- Difficult to extend with advanced PMU features.
- **High maintenance cost.**

Bugs in host PMU will cause accuracy and performance issue for vPMU

Counter state management highly depends on host PMU. A hard performance/accuracy compromise.

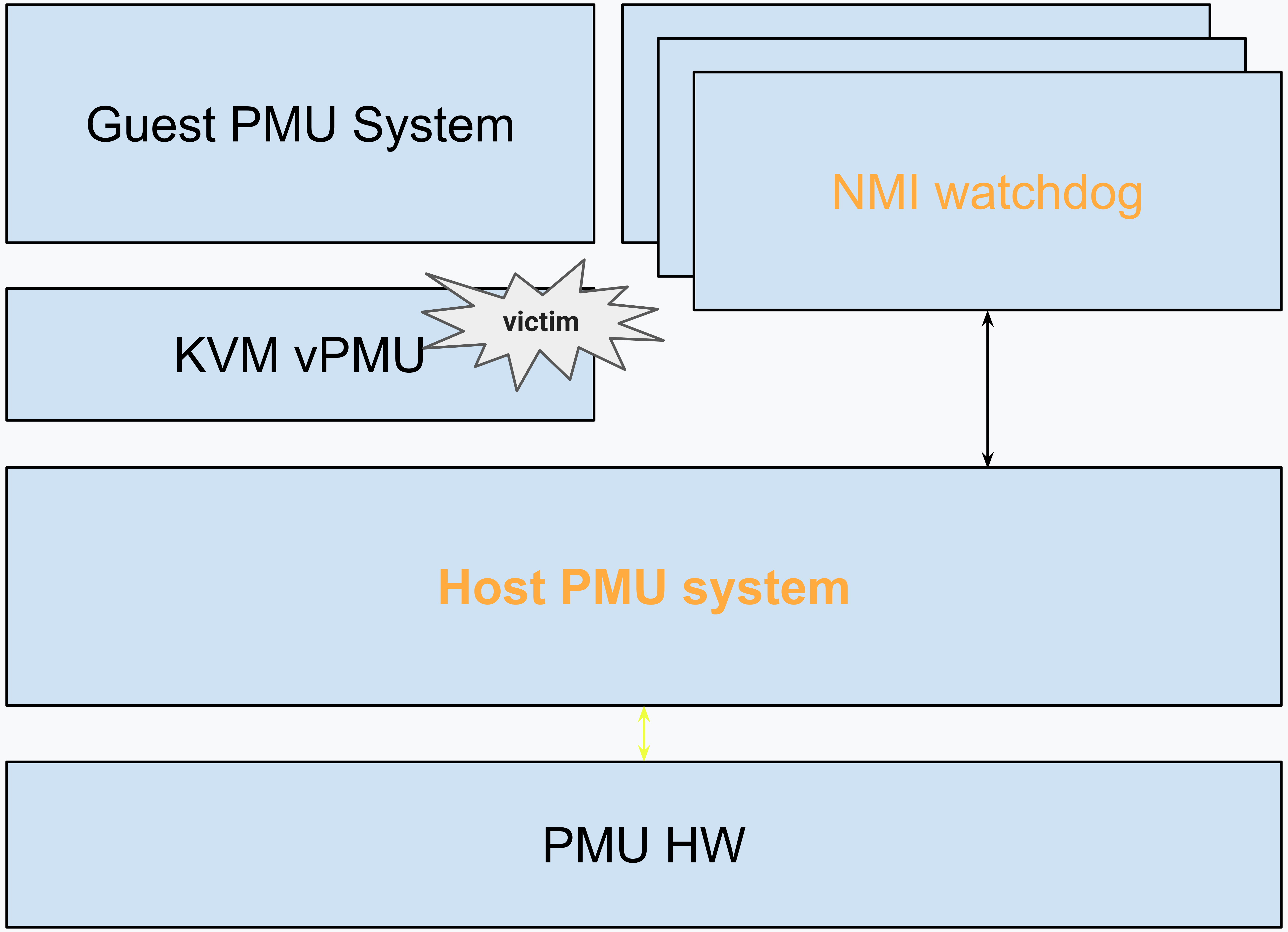


Existing vPMU in KVM

Existing vPMU implementation has the following issues

- High performance overhead
- Difficult to extend with advanced PMU features.
- High maintenance cost.
- **Silent errors**

Host PMU system may silently preempt the counters away from KVM vPMU



Google

Agenda

- Introduction to vPMU
- Motivation of Passthrough
- Design of Mediated Passthrough vPMU
- Current Status and Next Steps
- Q&A

Motivation of Mediated Passthrough vPMU

Passthrough vPMU allows guest with direct HW PMU access/ownership

Aims to solve all of the existing issues in current implementation

- Low performance overhead due to
    - passthrough accesses to PMU MSRs
    - being independent from host PMU system
- Easy to extend with new features:
    - LBRs, PEBS and Intel PTs, ...
- Easy to maintain
    - Code is simple.

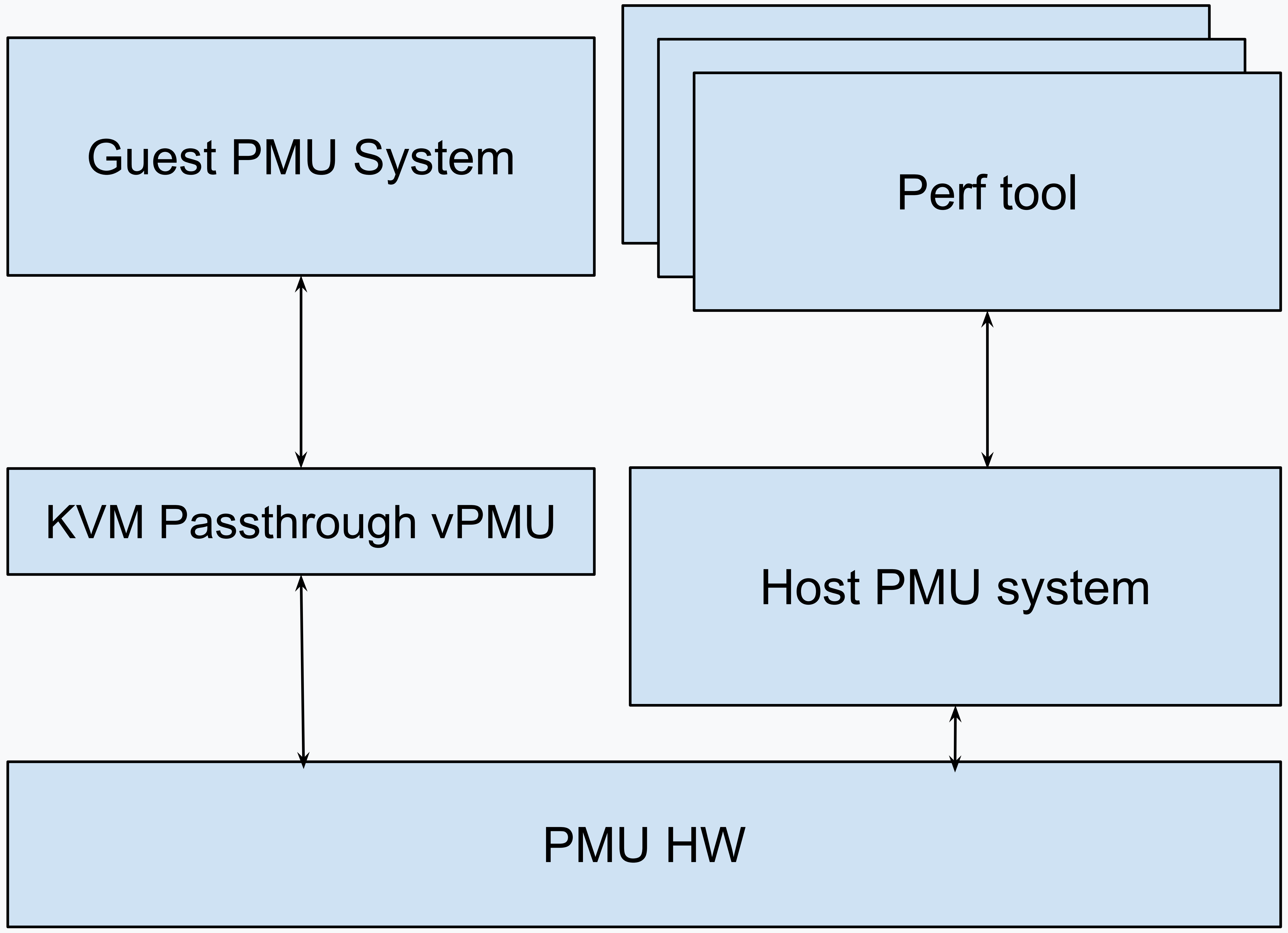
Agenda

- Introduction to vPMU
- Motivation of Passthrough
- Design of Passthrough vPMU
- Current Status and Next Steps
- Q&A

High-level Design

We made several high-level decisions on passthrough vPMU design

- Provide guest exclusive ownership of HW PMU
- KVM provides all necessary infra for PMU usage when guest is running



Low-level Decisions

Low-level decisions on passthrough vPMU includes

- Passthrough accesses to PMU MSRs except those to event selector MSRs
- Intercept RDPMC unless all counters are exposed to guest
- Use a separate PMI handler for KVM
- PMU context switch

Table 1. Interception of all PMU MSRs

	Interception	Reason
counters	**Pass through**	KVM vPMU fully owns all counters
selectors	**Intercept RW**	KVM check allowed events for security reason
rdpmc instruction	**Pass through**	if all counters are allowed for access
fixed counter ctrl	**Pass through**	performance
global ctrl	**Pass through**	performance
global overflow	**Pass through**	performance
global status	**Pass through**	performance

PMU Context Switch (global ctrl msr)

When control flows enter KVM from guest, Guest PMU counters has to stop. Otherwise, counter values are inaccurate.

Intel CPU

- Always Switch Global Ctrl MSR at VM Enter/Exit
  - For CPUs without VM_EXIT_{SAVE,LOAD}_IA32_PERF_GLOBAL_CTRL, use msr autosave/autorestore list to swap Global Ctrl MSR.

AMD CPU

- No automatic Global Ctrl MSR save/restore mechanism at VM Enter/Exit Boundary.
- Trick: when guest writes to event selectors:
  - Remove AMD64_EVENTSEL_HOSTONLY bit
  - Assign AMD64_EVENTSEL_GUESTONLY bit

PMU Context Switch (the rest)

PMU context switch is when switching ownership of HW PMU.

PMU context switch location	Pro	Con
Option #1: VM Enter/Exit	Host / Guest boundary is accurate and clear.	Higher overhead on VMExit 20+ MSR read/writes
Option #2: vCPU Loop	Better performance. Eg., almost no overhead of slice of HW	Host loses the capability of profiling KVM run loop.

Always Sync HW APIC_LVTPC mask with KVM virtual APIC at VM Enter/Exit

Agenda

- Introduction to vPMU
- Motivation of Passthrough
- Design of Passthrough vPMU
- Current Status and Next Steps
- Q&A

Drawbacks and Open Discussions

- **Host profiling limitation**: Host users lose the ability to profile guests when the new vPMU mode is enabled.
- **NMI watchdog**: Perf event for NMI watchdog is affected, potential solutions are being explored (buddy hardlock detector, HPET-based hardlock detector).
- **Dedicated kvm_pmi_vector**: Ensures correct PMI handling in passthrough vPMU, avoiding confusion between host and guest PMIs.
- **PMU context switch location**: Debate on whether to perform at VM Enter/Exit (current) or VCPU_RUN loop boundary (alternative), considering performance overhead and profiling capabilities.

Next Steps

Optimizations on PMU Context Switches

- Lazy PMU Context Switch in KVM
- Enlightened PMU Context Switch around VM-enter/exit

Mediated Passthrough vPMU for KVM

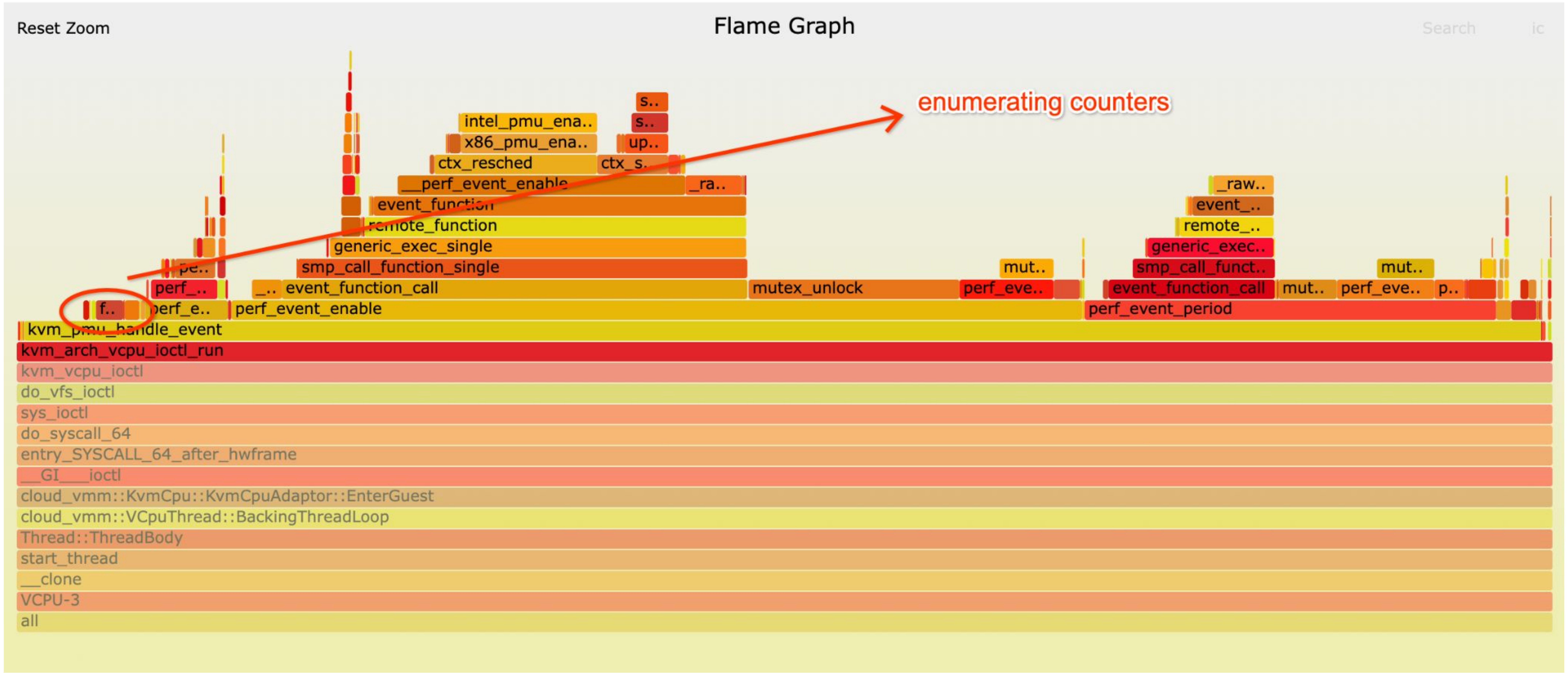
Q & A

Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024

Majority of the time is wasted in

- perf_event_enable()
- perf_event_period()
- perf_event_create_kernel_counter()
- **find_next_bit()**



Google