Virtual machine monitoring using trace analysis

Mohamad Gebai
Michel Dagenais

11 December, 2013
École Polytechnique de Montreal

Content

- General objectives
- TMF – Virtual Machine View
- Trace synchronization
- Other work in progress

General objectives

- Getting the state of a virtual machine at a certain point in time
- Quantifying the overhead added by virtualization
- Track the execution of processes inside a VM
- Aggregate information from host and guests
- Monitoring multiple VMs on a single host OS
- Building a state system in TMF for virtual machine support
- Finding performance setbacks due to resource sharing among VMs

Tracing

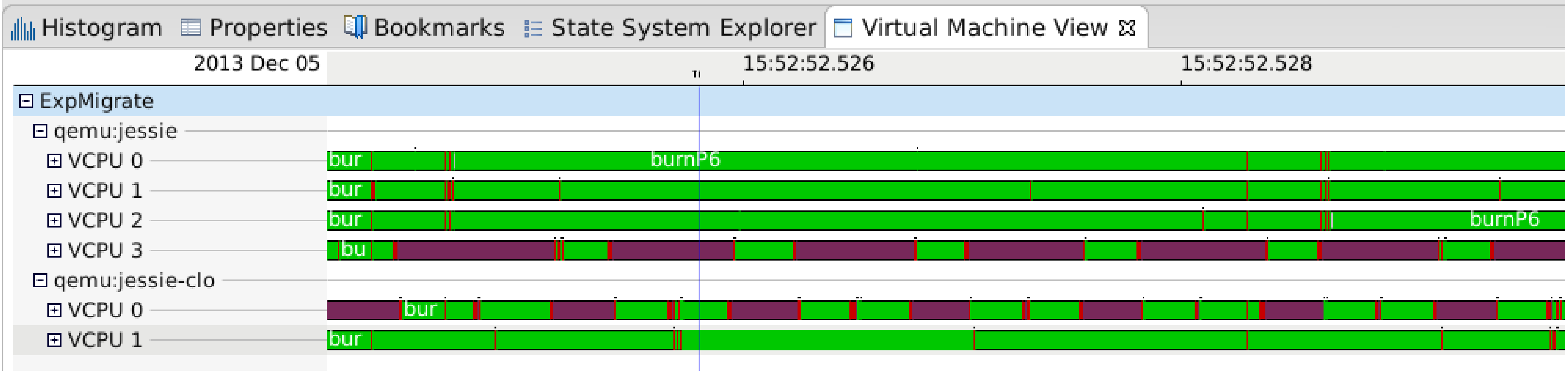
- Using LTTng for kernel tracing
- KVM as a hypervisor
- Trace scheduling events
    - sched_switch for context switching
    - sched_migrate_task for thread migration between CPUs
- Trace system calls (optional)
- Trace interrupts (optional)
- Qemu userspace tracing (optional)
- Trace VMENTRY and VMEXIT on the hypervisor (hardware virtualization)

Tracing virtual machines

- Each vCPU is 1 thread
- A vCPU can be in VMX root mode or VMX non-root mode
- A vCPU can be preempted on the host
- The VM can not know when it is preempted or in VMX root mode
- Processes in the VM seem to take more time

TMF Virtual Machine View

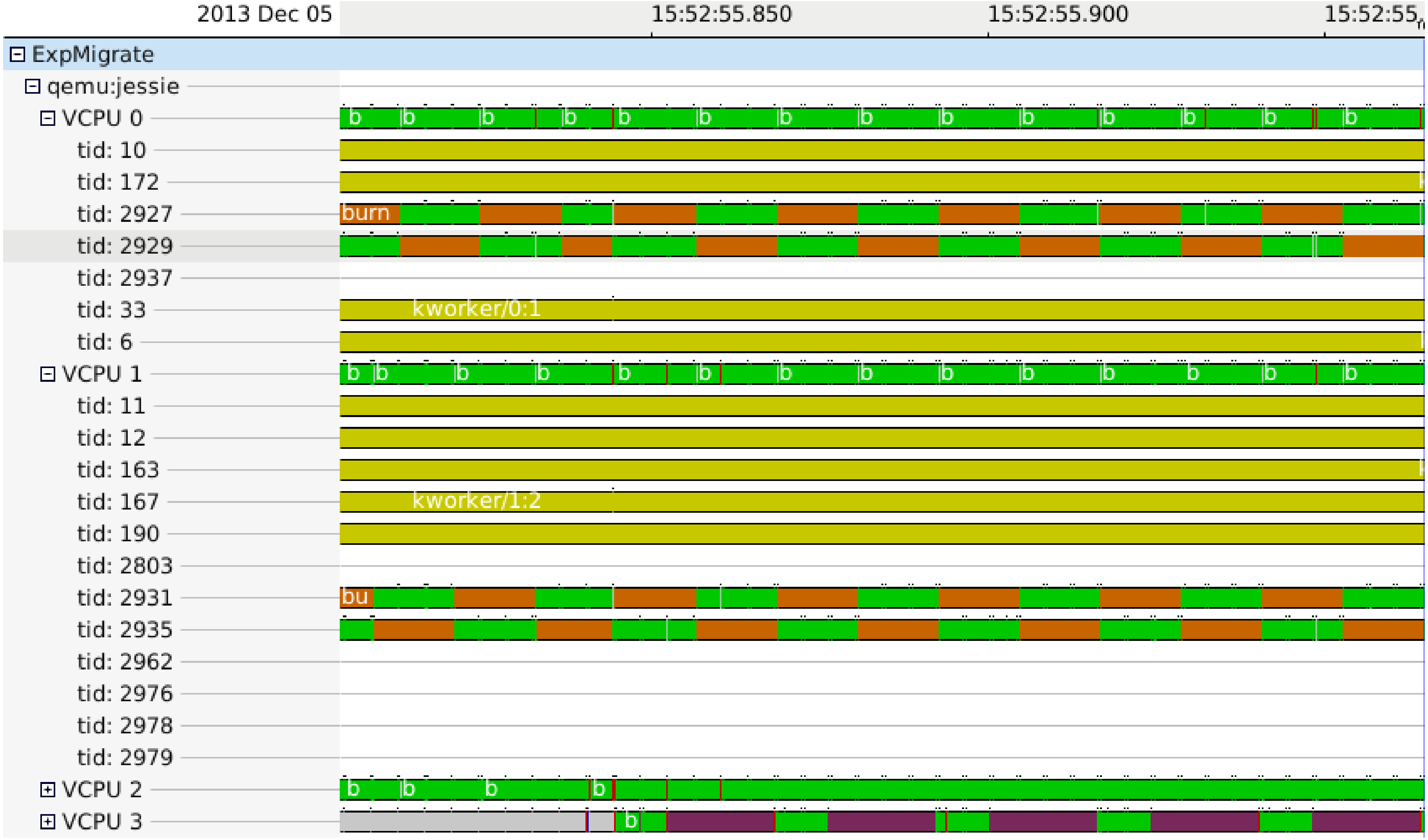
- Shows the state of each vCPU of a VM
- Aggregation of traces from the host and the guests



- 2 VM:
  - Jessie: 4 vCPUs
  - Jessie-clone: 2 vCPUs
  - vCPU 3 and vCPU 0 are complementary

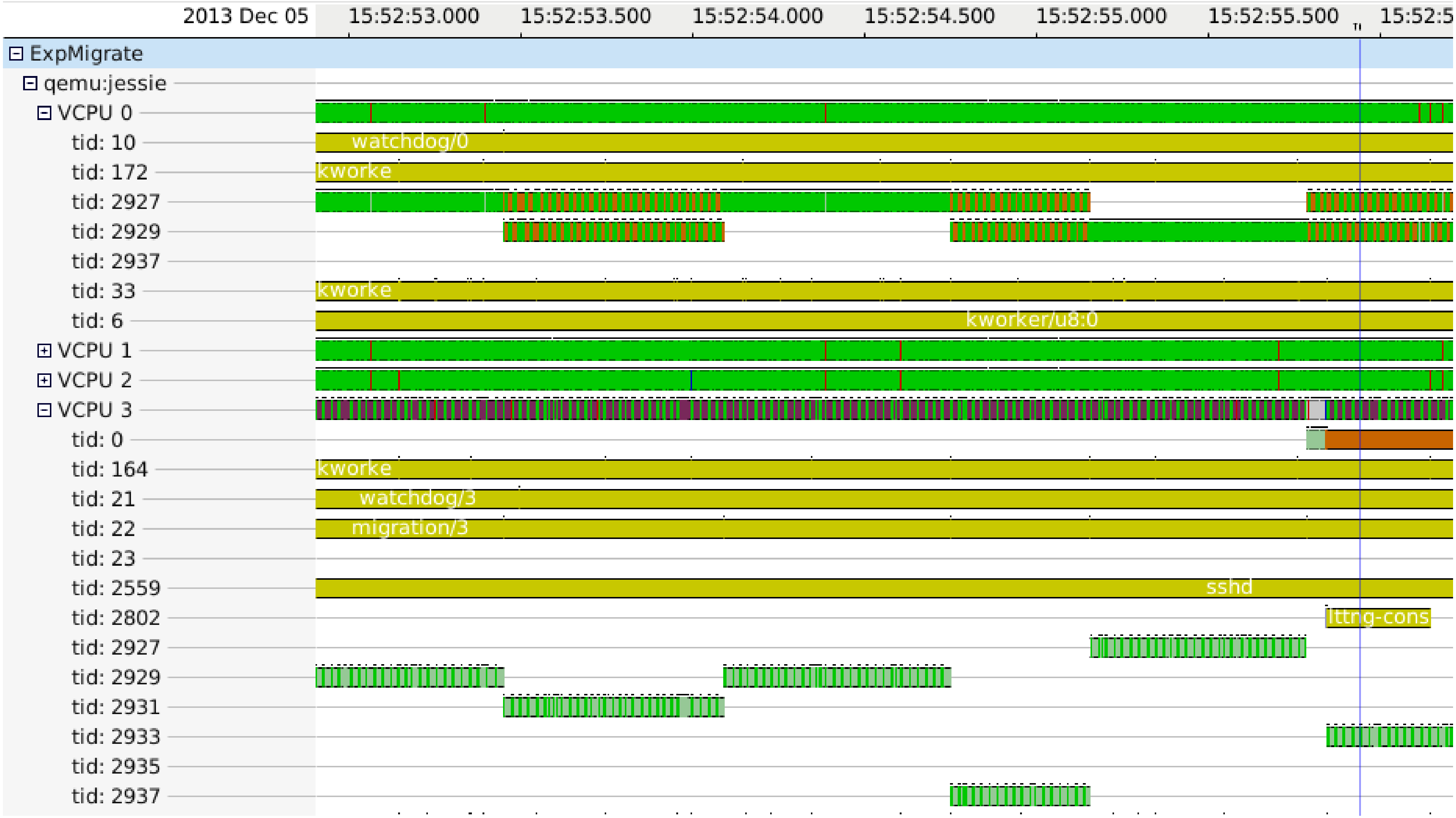
TMF Virtual Machine View

- Shows execution details inside the VM



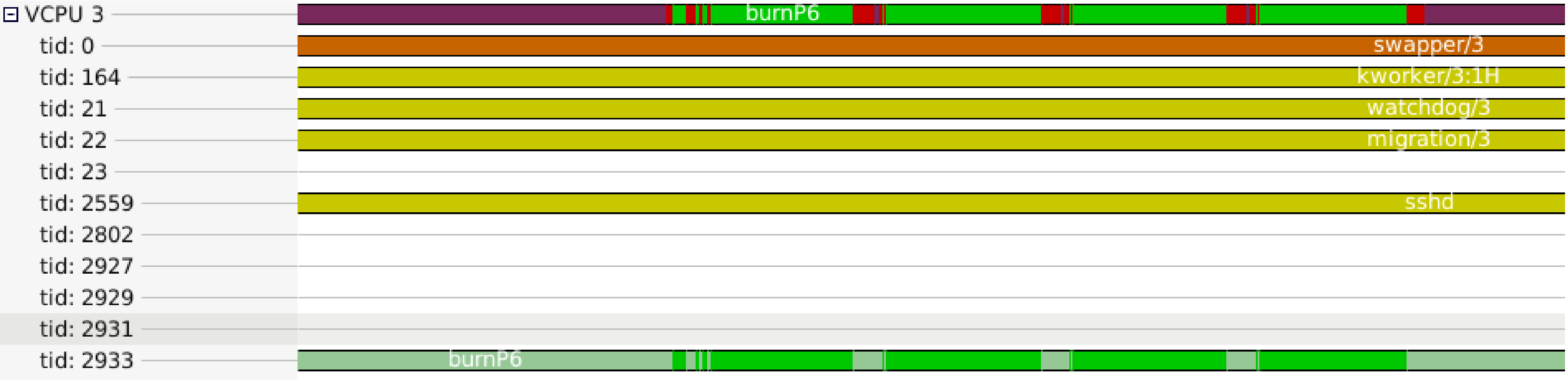
TMF Virtual Machine View

- Shows information about processes and task migration



TMF Virtual Machine View

- Shows latency introduced by the hypervisor and by vCPU preemption
  - vCPU:
    - Red: hypervisor code
    - Green: user mode
    - Purple: vCPU preempted
  - Threads:
    - Green: user mode
    - Grey: thread appears to be running for the guest but is actually preempted



Trace synchronization

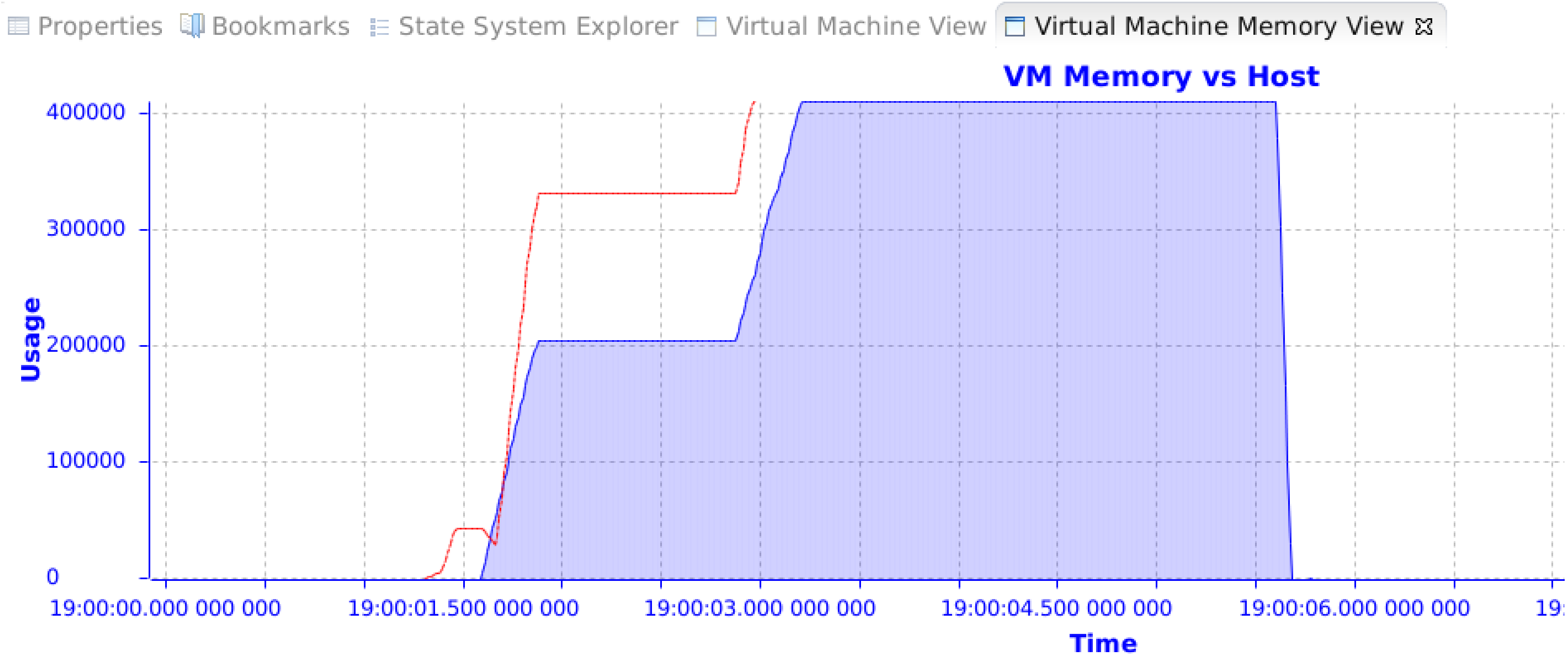
- Based on the fully incremental convex hull synchronization algorithm
- 1-to-1 relation required between events from guest and host
- Tracepoint is added to the guest kernel: trace_periodic_hypercall(counter)
- Executed on the system timer interrupt softirq
- This tracepoint triggers a hypercall which is traced on the host: trace_kvm_hypercall(counter)
- Requires hardware-assisted virtualization for the hypercall instruction
- Resistant to VM migrations, vCPU migrations and time drifts

Memory usage

- Upon creation, a VM allocates its total RAM in memory
- Pages are actually allocated when touched by processes inside the VM
- When pages are freed inside the VM, the memory is not freed on the host
- Solution: ballooning (Kernel thread which allocates memory and gives it back to the host)
  - Ballooning is done by defining rules
  - Ex: 80% of memory of the VM is used
  - VM will start swapping
- These rules do not guarantee to choose the best VM for ballooning
  - KSM (Kernel Samepage Merging)
  - Ex: 20% of memory of the VM is used, but previous peak of 90% → 70% of unused allocated frames

Memory usage

- Trace page allocation and page freeing on the host and the guest



Future work

- Instrument KSM
- Redefine rules for ballooning while taking into consideration KSM and unused touched frames

Acknowledgement

- Thanks to Genevieve Bastien for her help in TMF