Identifying and reducing virtualization overhead

François Belias

2 February 2025

Polytechnique Montréal

DORSAL Laboratory

Agenda

Motivation

- What is virtualization ?
- Why detecting virtualization overhead is important ?
- Research objectives

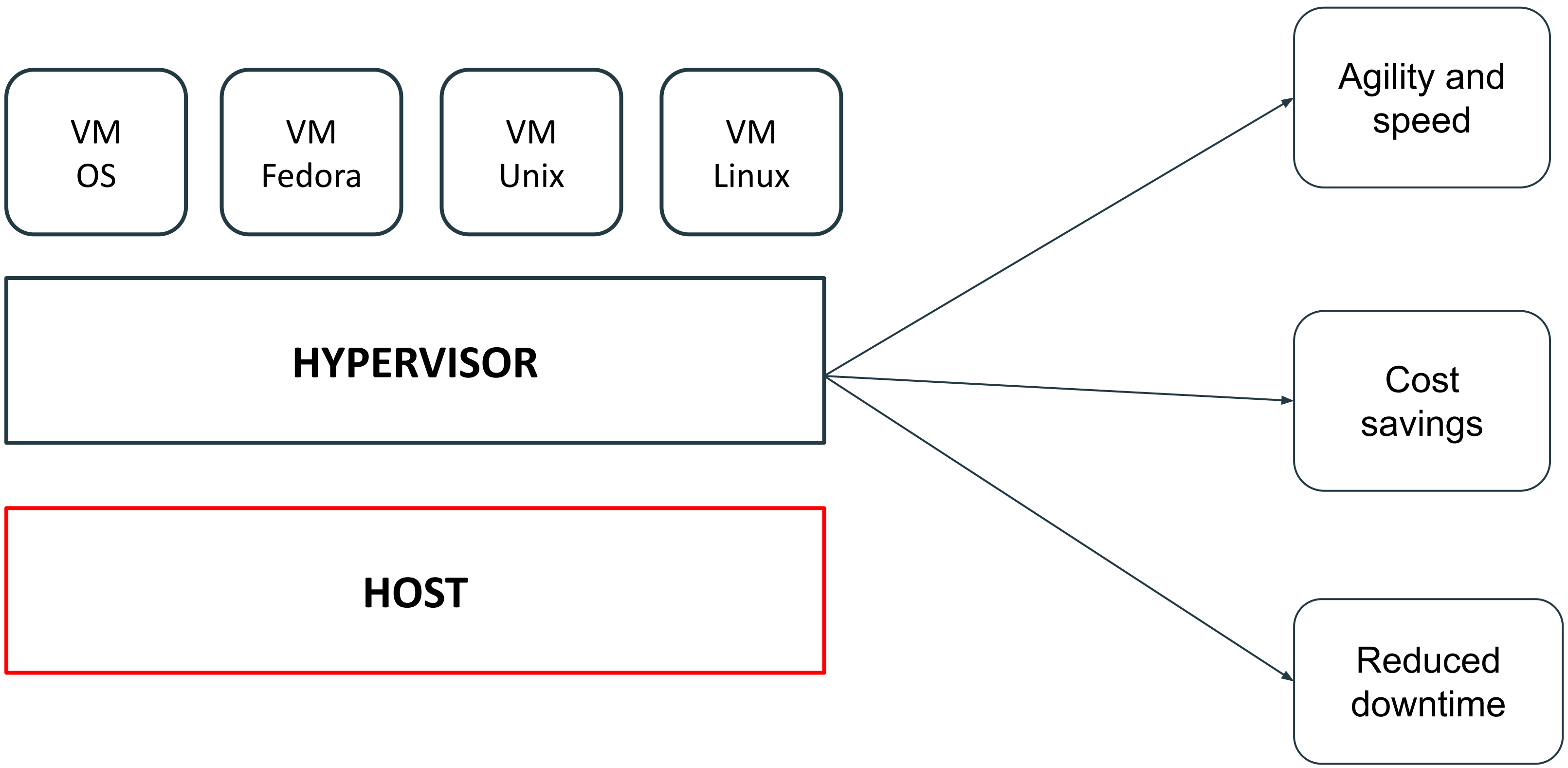
Update

- Integration of analyzes into trace compass
- New feature: Execution comparison for Native system vs virtualized system
- Workload tracing results

Conclusion and in-progress

Motivation

What is virtualization ?



VM
OS

VM
Fedora

VM
Unix

VM
Linux

HYPERVISOR

HOST

Agility and speed

Cost savings

Reduced downtime

Motivation

Why detecting virtualization overhead is important ?

- **Performance optimization**
  - Virtualization overhead can lead to unnecessary ressources consumption, which might affect the performance of virtual machines
  - By detecting this overhead, administrators can make adjustments to ensure optimal performance.
  - Detection of performance bottlenecks caused by the virtualization layers itself to avoid performance degradation
- **Cost efficiency**
  - Detecting and minimizing virtualization overhead can improve the cost-efficiency of the environment by avoiding wasted resources
- **Troubleshooting and Diagnosis**
  - Detecting whether virtualization overhead is a contributing factor can help with diagnosing the problem quickly

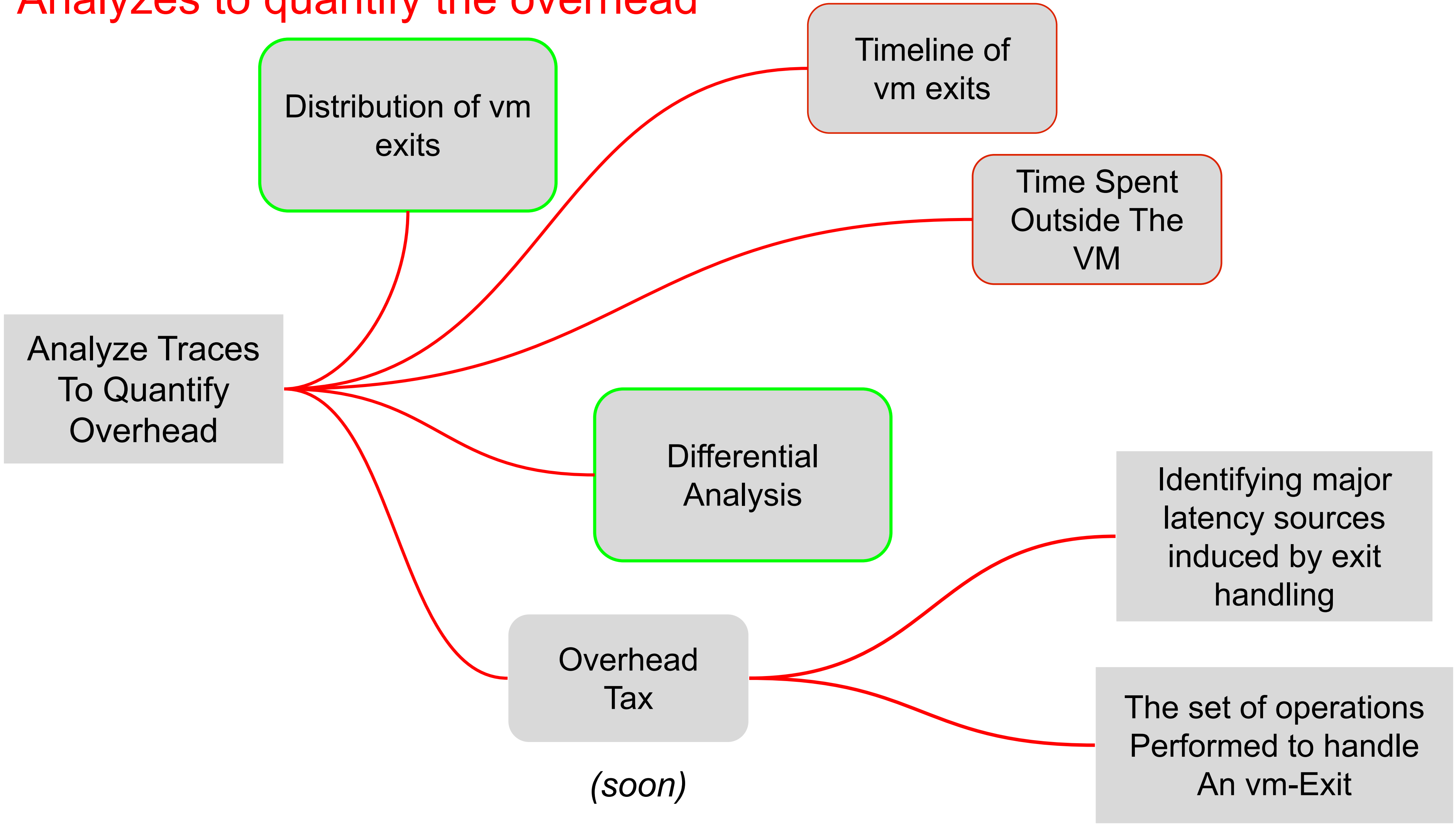
Motivation

Research objectives

- Provide practitioners with analyses, tools, or approaches to identify the causes of the “additional overhead” introduced by virtualization (VM overhead).
- Enhance existing performance analysis tools, such as Trace Compass, LTTng, or related tools.

Methodology

Analyzes to quantify the overhead



Analyze Traces To Quantify Overhead

Distribution of vm exits

Timeline of vm exits

Time Spent Outside The VM

Differential Analysis

Overhead Tax

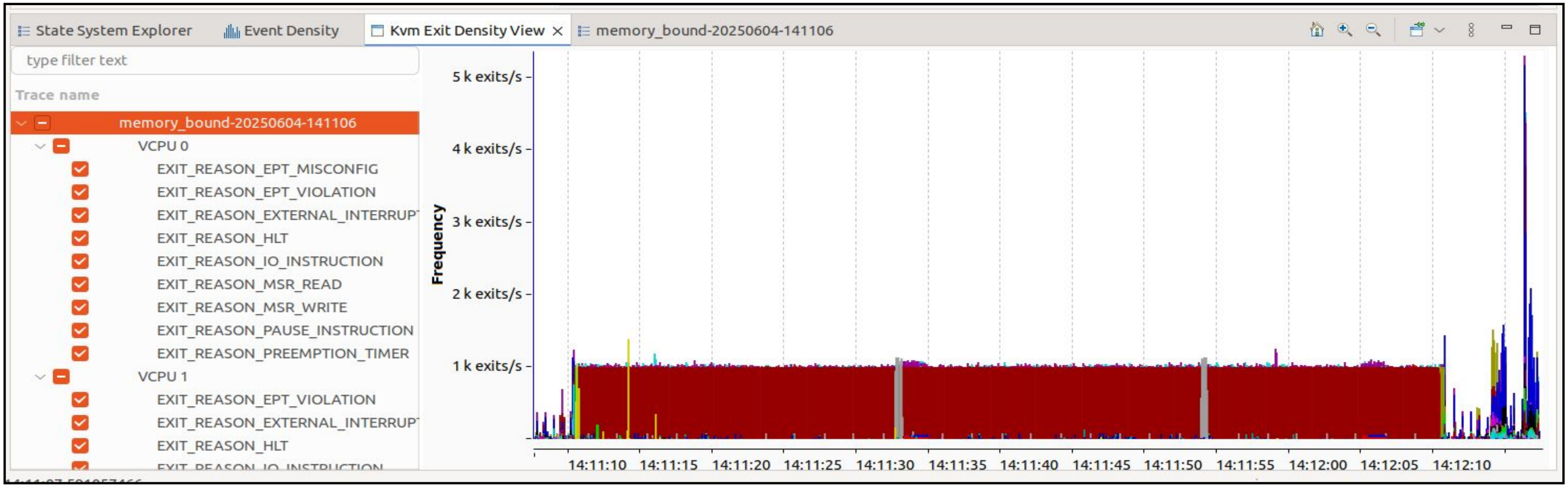
(soon)

Identifying major latency sources induced by exit handling

The set of operations Performed to handle An vm-Exit

Update

Integration of the distribution of KVM exits into trace compass

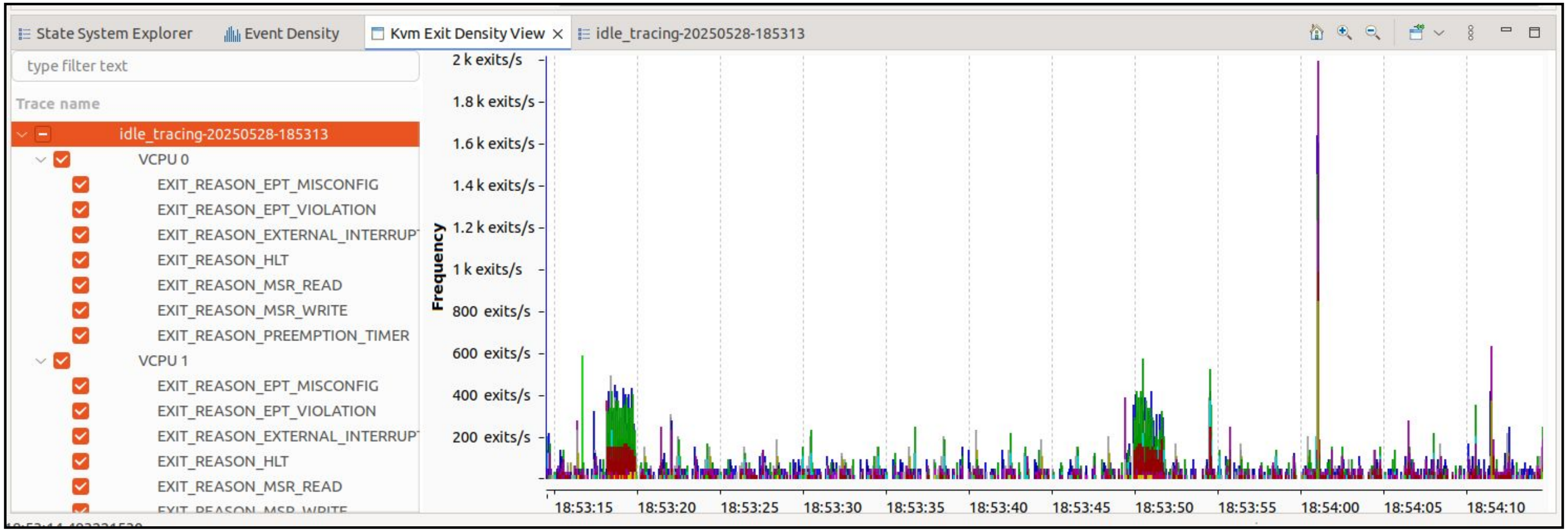


- memory_bound-20250604-141106
  - VCPU 0
    - ☑ EXIT_REASON_EPT_MISCONFIG
    - ☑ EXIT_REASON_EPT_VIOLATION
    - ☑ EXIT_REASON_EXTERNAL_INTERRUPT
    - ☑ EXIT_REASON_HLT
    - ☑ EXIT_REASON_IO_INSTRUCTION
    - ☑ EXIT_REASON_MSR_READ
    - ☑ EXIT_REASON_MSR_WRITE
    - ☑ EXIT_REASON_PAUSE_INSTRUCTION
    - ☑ EXIT_REASON_PREEMPTION_TIMER
  - VCPU 1
    - ☑ EXIT_REASON_EPT_VIOLATION
    - ☑ EXIT_REASON_EXTERNAL_INTERRUPT
    - ☑ EXIT_REASON_HLT
    - ☑ EXIT_REASON_IO_INSTRUCTION

Temporal view of KVM exit distribution for memory-bound application

Update

Integration of the distribution of KVM exits into trace compass



- ☐ idle_tracing-20250528-185313
  - ☑ VCPU 0
    - ☑ EXIT_REASON_EPT_MISCONFIG
    - ☑ EXIT_REASON_EPT_VIOLATION
    - ☑ EXIT_REASON_EXTERNAL_INTERRUPT
    - ☑ EXIT_REASON_HLT
    - ☑ EXIT_REASON_MSR_READ
    - ☑ EXIT_REASON_MSR_WRITE
    - ☑ EXIT_REASON_PREEMPTION_TIMER
  - ☑ VCPU 1
    - ☑ EXIT_REASON_EPT_MISCONFIG
    - ☑ EXIT_REASON_EPT_VIOLATION
    - ☑ EXIT_REASON_EXTERNAL_INTERRUPT
    - ☑ EXIT_REASON_HLT
    - ☑ EXIT_REASON_MSR_READ
    - ☑ EXIT_REASON_MSR_WRITE

Temporal view of KVM exit distribution for an Idle VM

Differential Performance Analysis: Virtualization Overhead in Real-world workloads

Where does virtualization overhead come from in real applications ?

Current limitations:

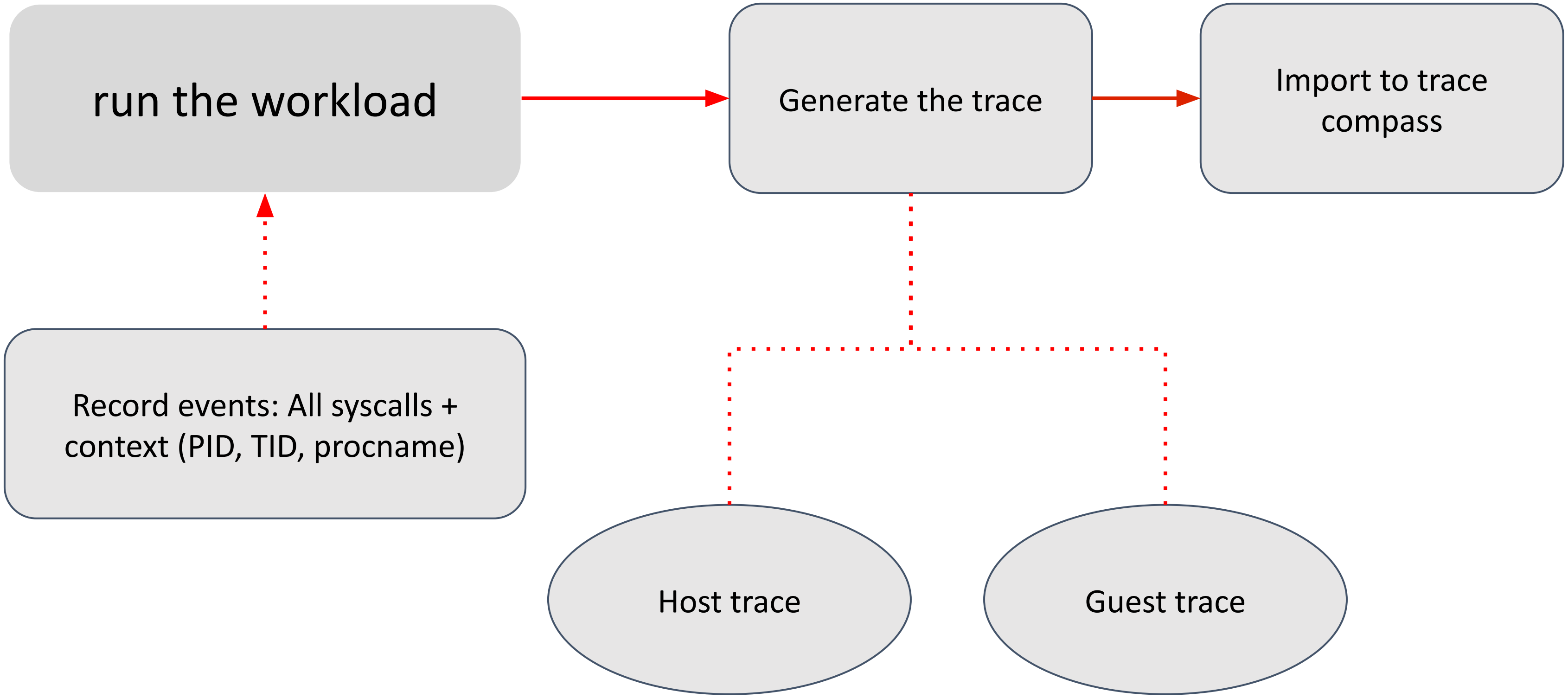
- ❌Macro level metrics (throughput, latency)
- ❌No visibility into system call overhead
- ❌Hard to pinpoint optimization opportunities

Our approach:

- ✅Differential kernel tracing (native vs VM)
- ✅System call-level analysis
- ✅Multi-process tracking

Methodology

Tracing approach



Workloads selection

Two representative workloads:

Sysbench

❏ Category: I/O intensive

❏ use case: Writing 1GB file on the disk

❏ Characteristics: Sequential writes

ML Training using pytorch on cpu

❏ Category: CPU intensive

❏ Use case: AI/ML workload

❏ Characteristics: Compute-heavy, minimal I/O

Goal: Cover different overhead patterns

Workload tracing results

https://drive.google.com/file/d/1F17z6377bdLYs3aCwkTFy6EUGvzb8ldH/view?usp=sharing

Conclusion and in-progress

Summary

- Integration of the KVM exit analysis into trace compass
- It is possible to correlate the guest activity and exits
- Introducing a comparison analysis using the VM and the native traces

Going further

- Test on more workload
- Integrate the analysis to provide context around exits (*overhead tax*)

Questions ?

Workload suggestions ? send a description to francois-philippe.ossim-belias@polymlt.ca