Identifying and reducing virtualization overhead

François Belias

12 April 2025

Polytechnique Montréal

DORSAL Laboratory

Agenda

Motivation

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

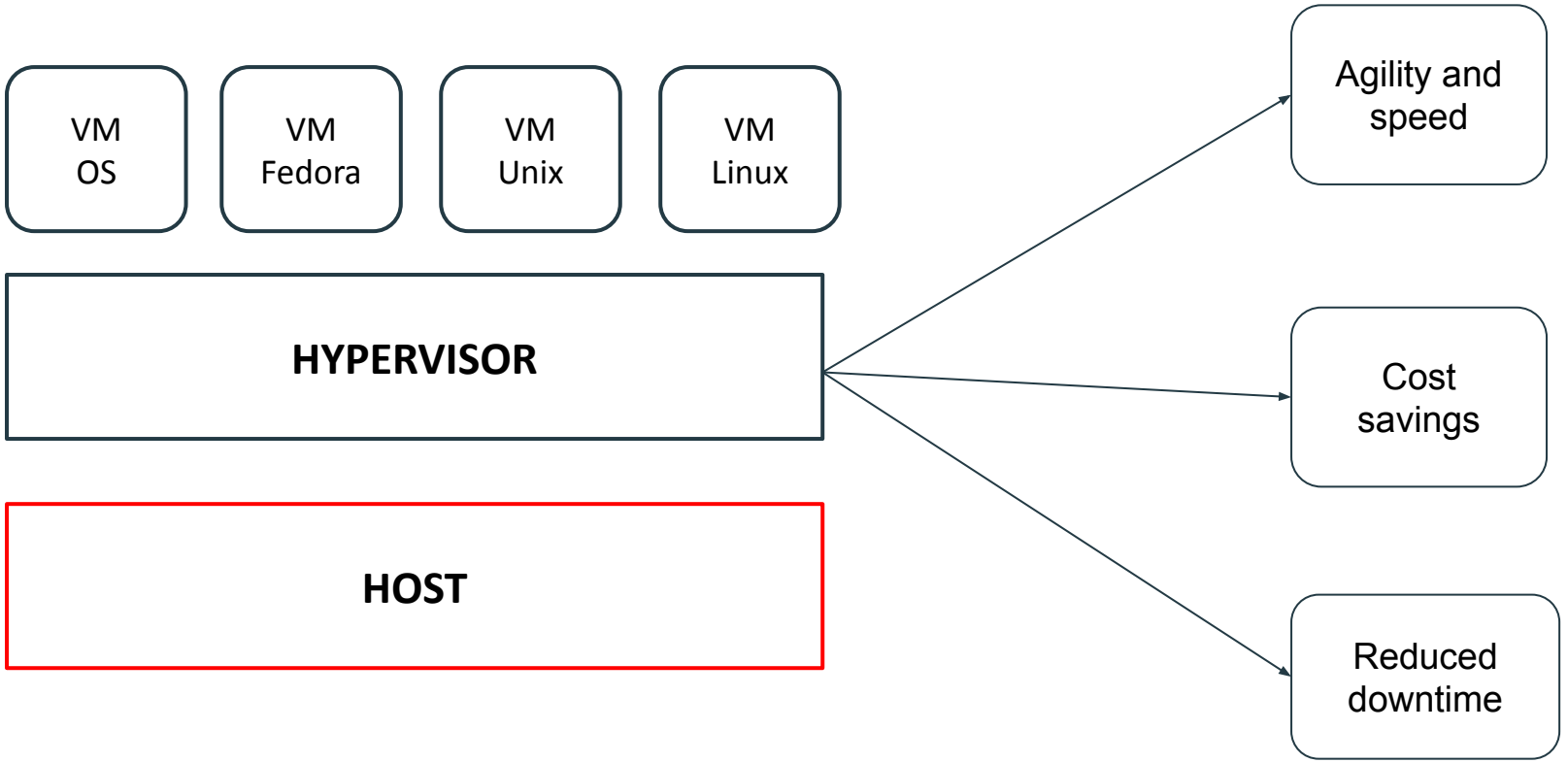
Methodology

- Find a workload that generates the overhead
- Tracing approach to collect data
- Analyzes to quantify the overhead

Conclusion and in-progress

Motivation

What is virtualization ?



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

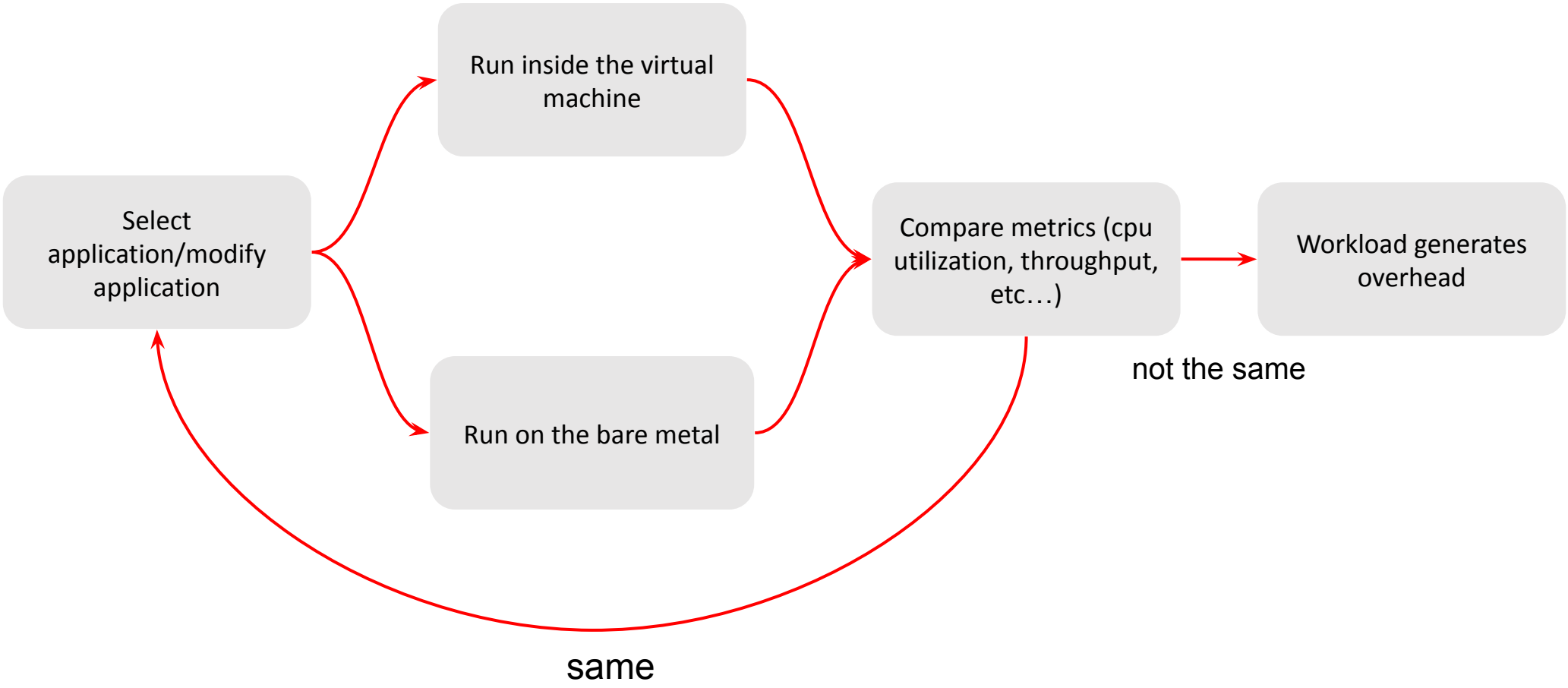
Motivation

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.

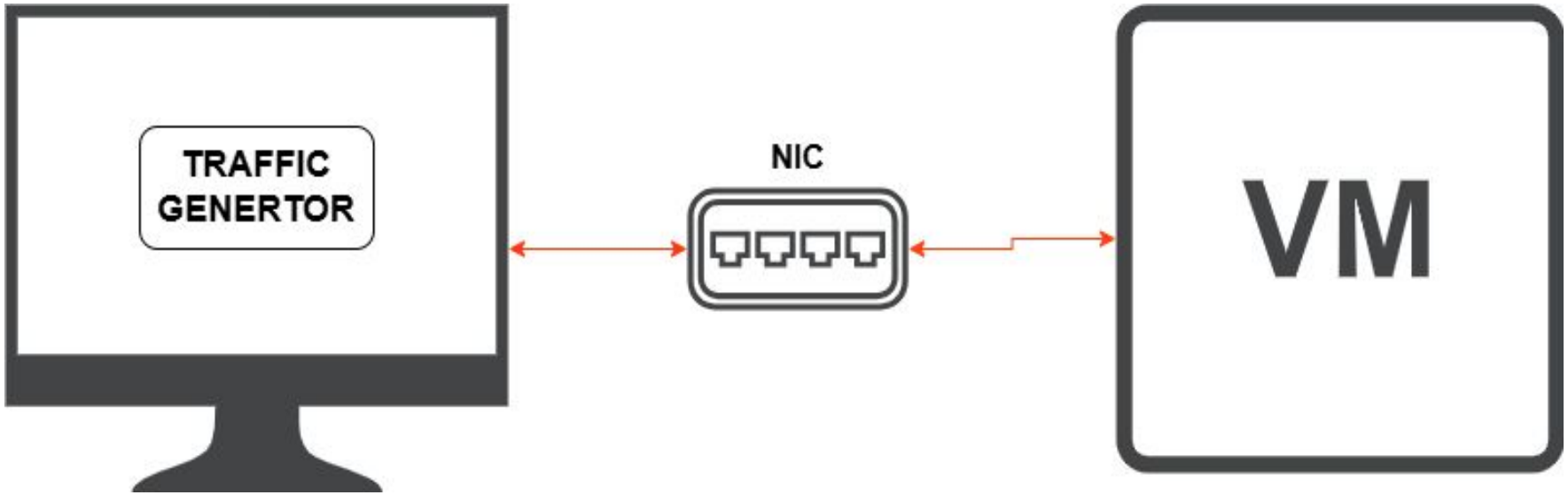
Methodology

Find workload that generate the overhead



Methodology

Find a workload that generate overhead



VM has direct access to the network interface (NIC) using SR-IOV

Methodology

Find a workload that generate overhead

- **Tools used**
  - TREX for traffic generation
  - DPDK test-pmd to process the packets and write informations about the packets on the disk
- **Tests configurations**
  - Without CPU pinning
  - With CPU pinning
- For each configuration we measure the **throughput** and **cpu utilization**
- We compare the results between the bare metal execution and the virtual machine execution

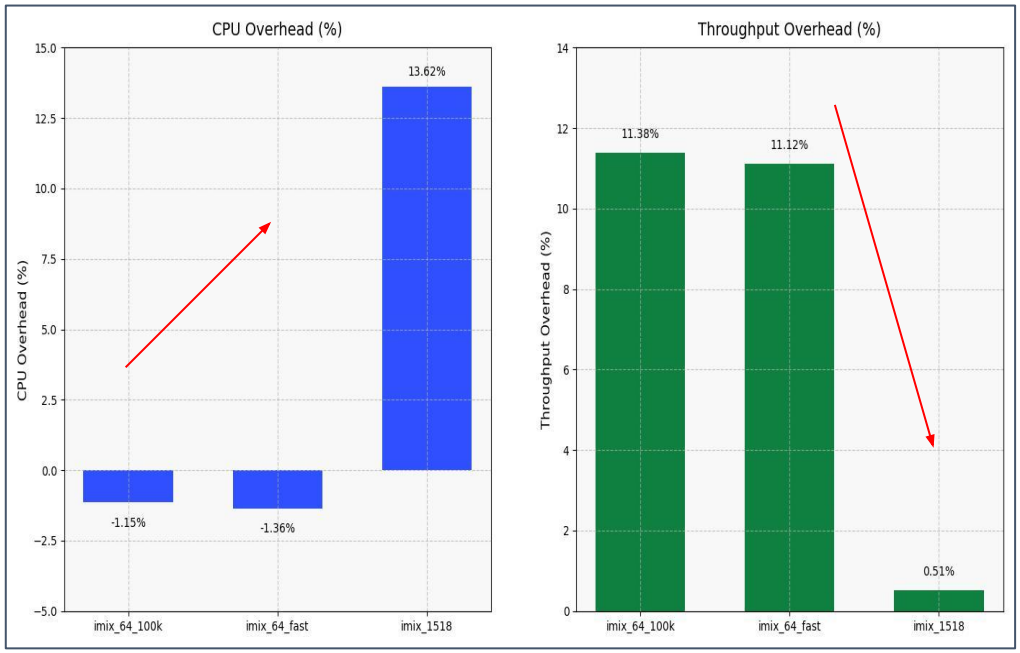
Methodology

Results of the benchmarks

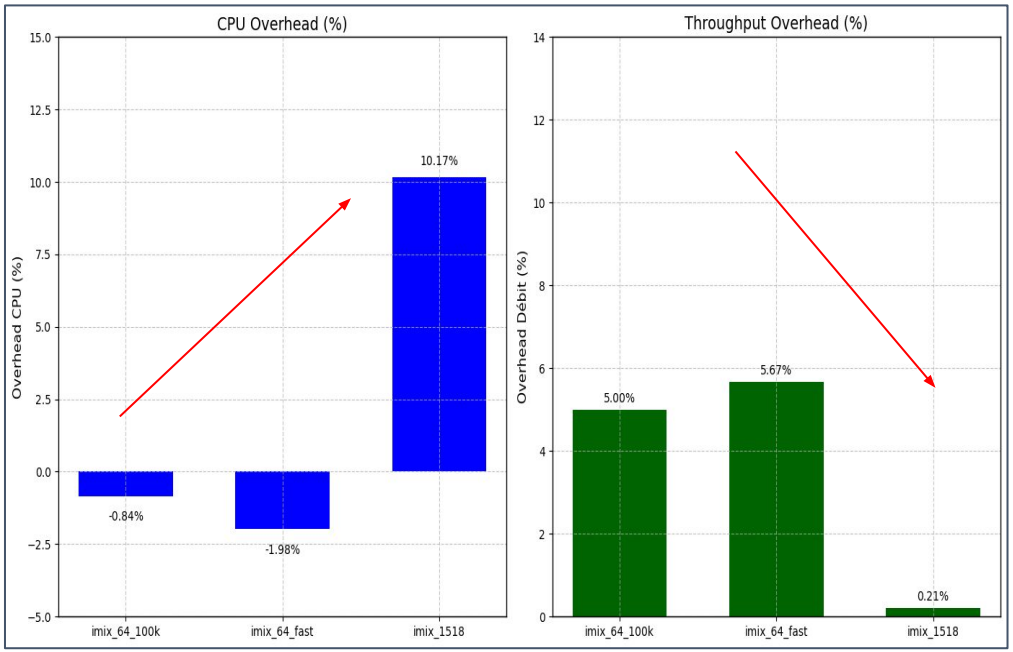
- Throughput overhead decreases with packet size
- CPU overhead increases with packet size
- CPU pinning increase the performance but the tendency is the same

Methodology

Results of the benchmarks



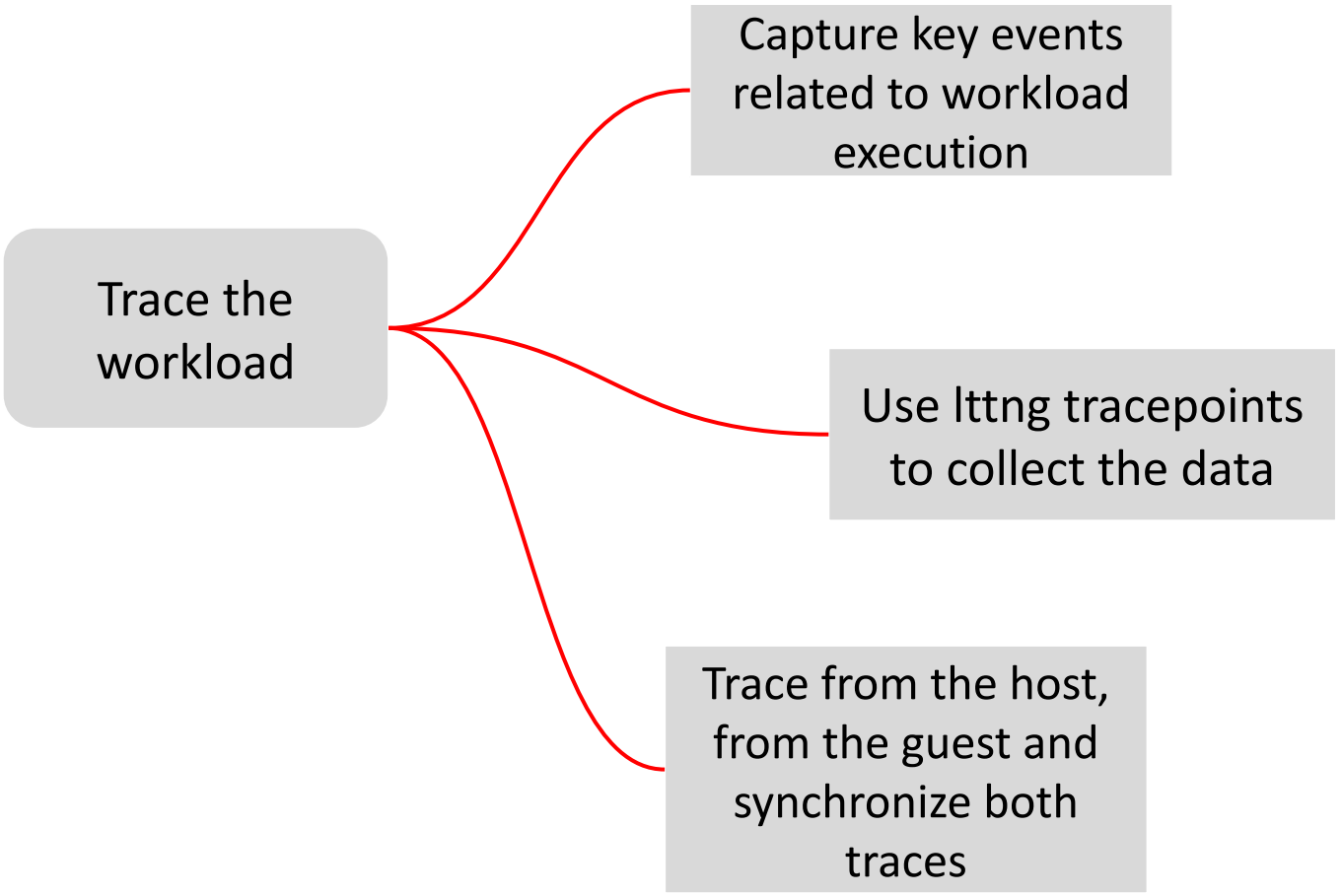
CPU and throughput overhead without cpu pinning



CPU and throughput overhead with cpu pinning

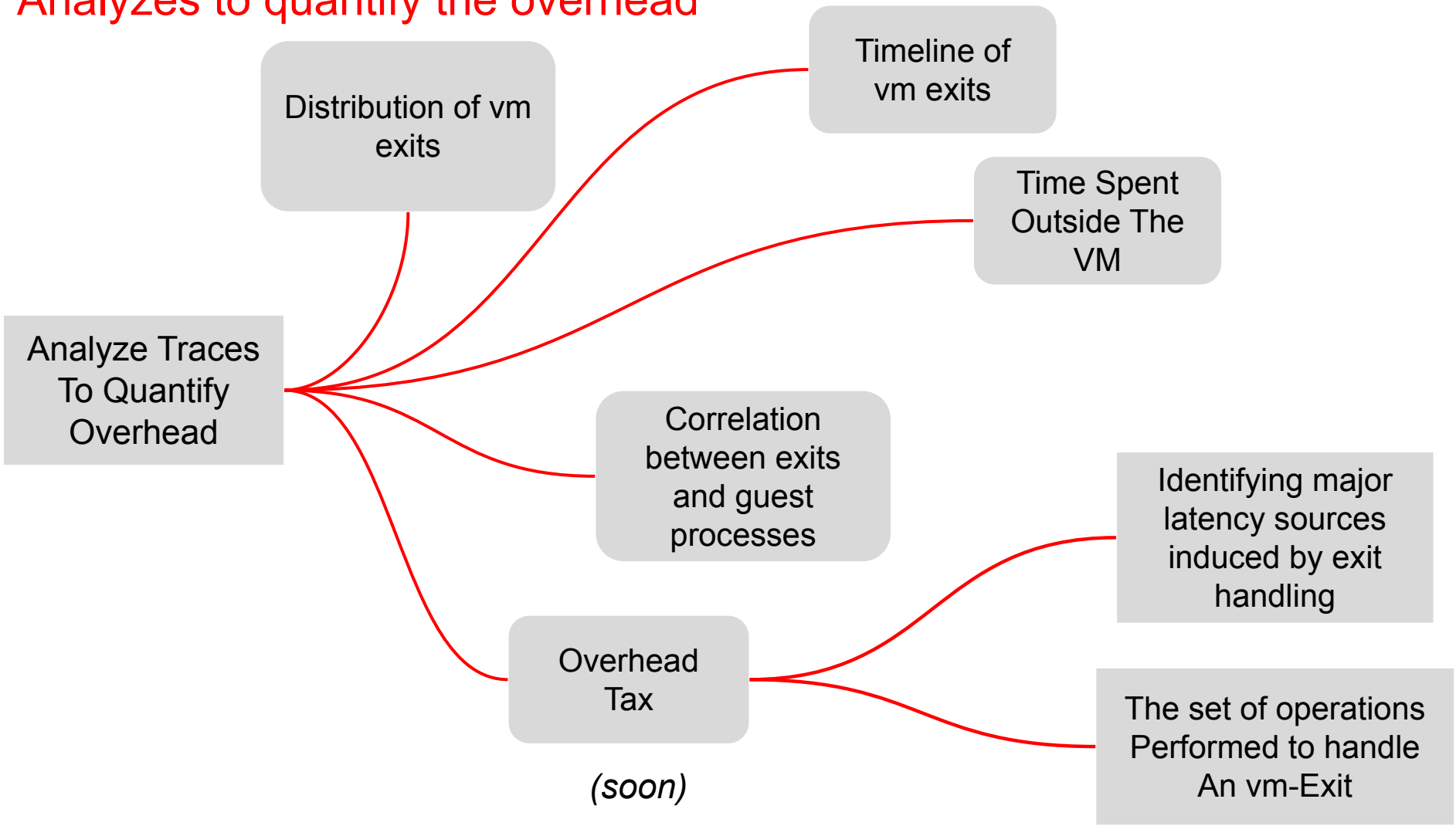
Methodology

Tracing approach



Methodology

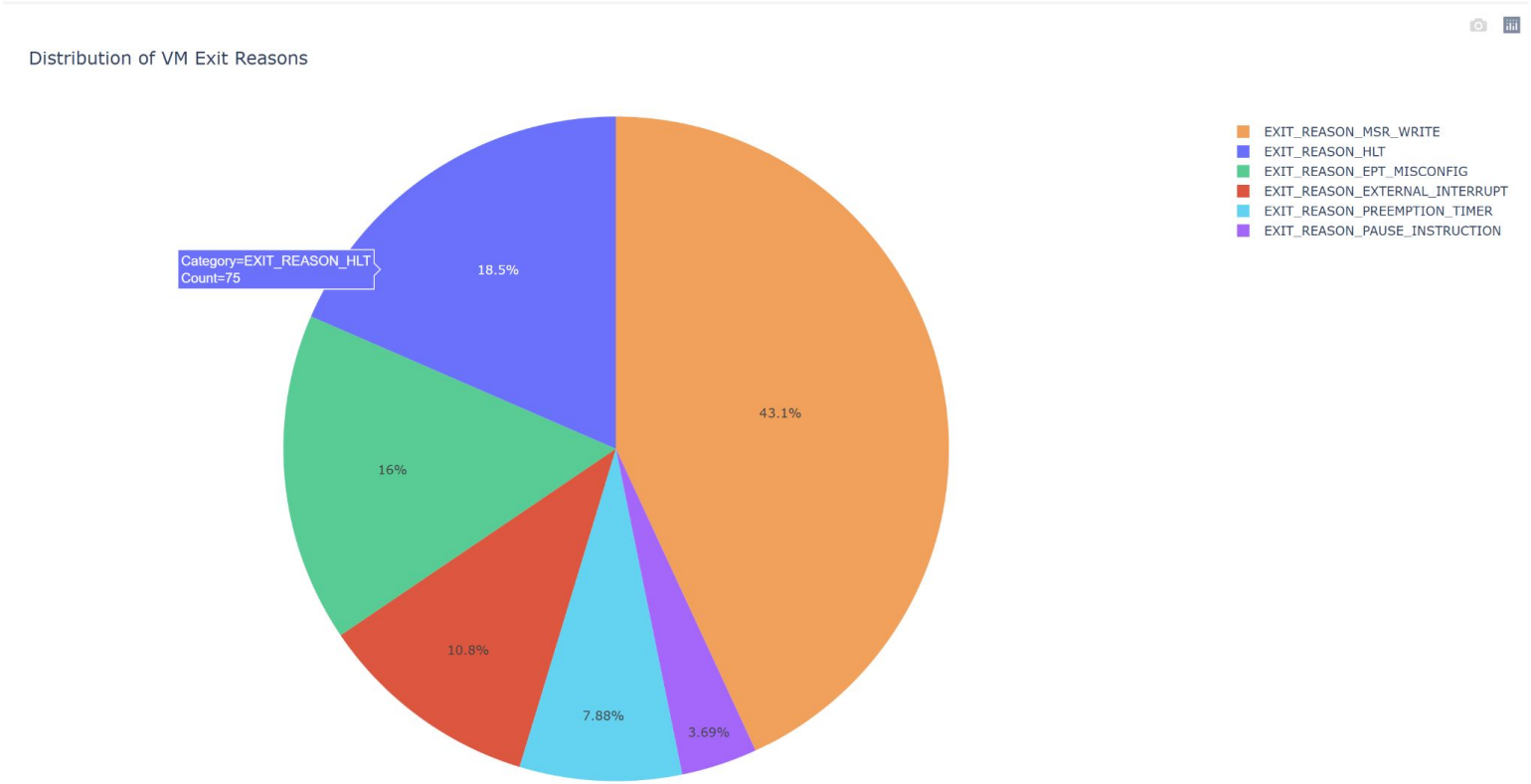
Analyzes to quantify the overhead



Analyzes to quantify the overhead

Distribution of vm exits

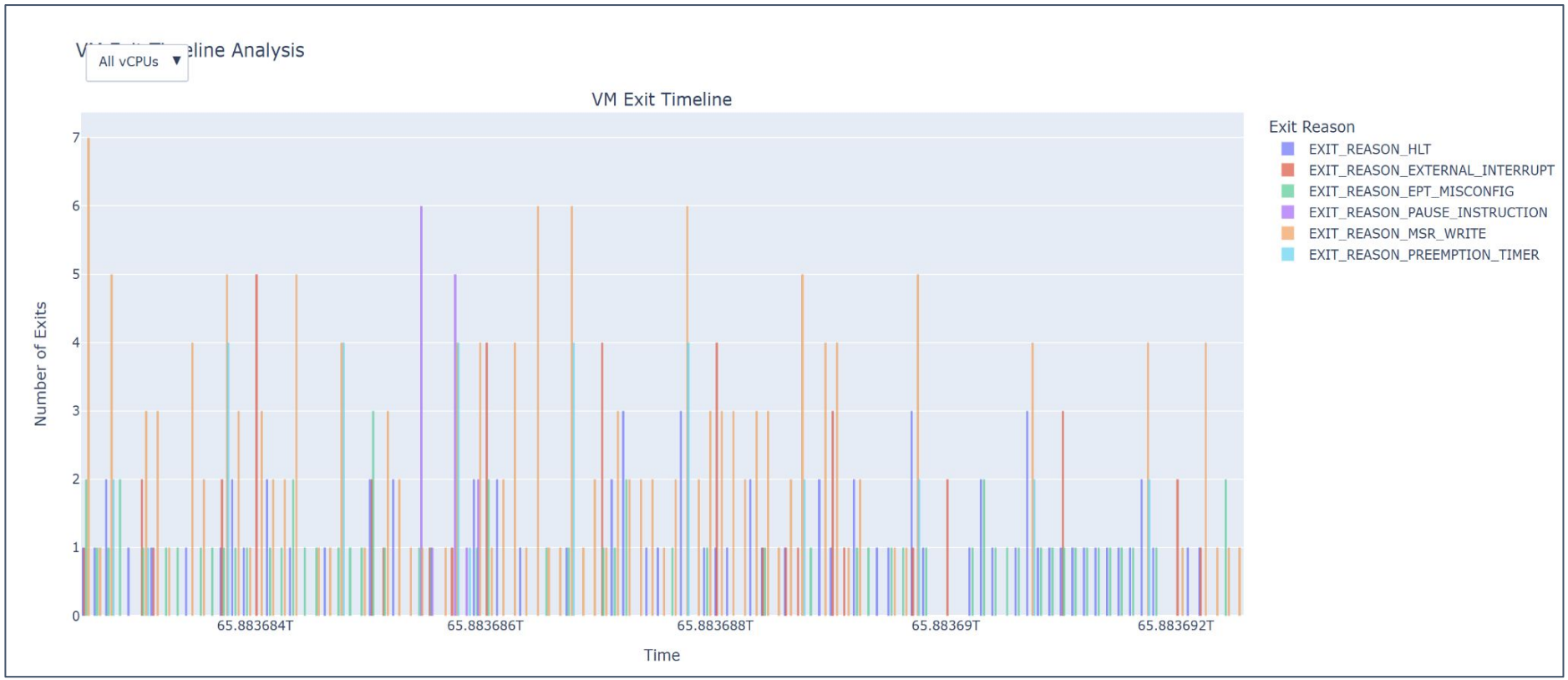
- It allows us to **see which exit types are most frequent**.
- It helps to **prioritize optimizations**: for example, if EXIT_REASON_IO_INSTRUCTION is dominant, it suggests that IO virtualization is a bottleneck.



Analyzes to quantify the overhead

Timeline of vm exits

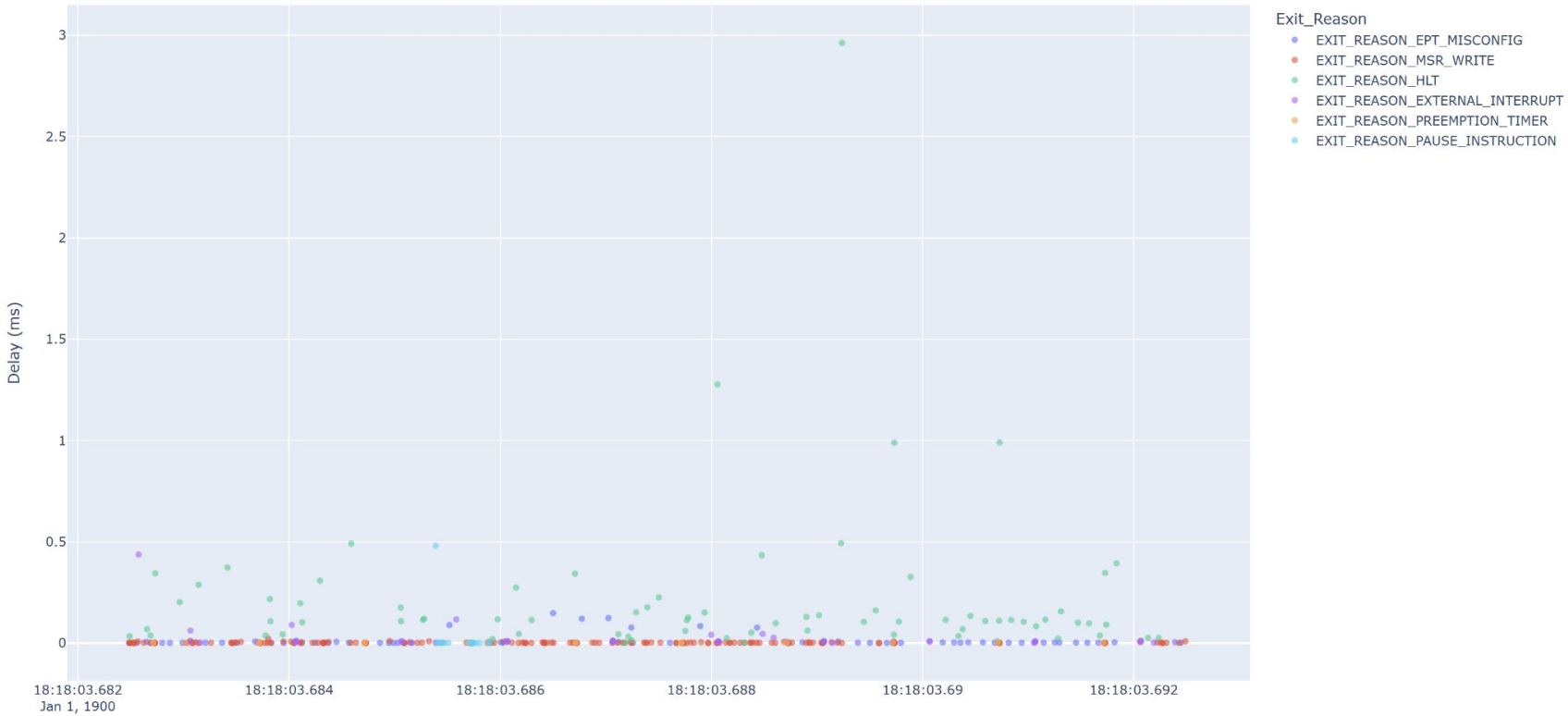
- It allows us to **see the evolution of exits** over time.
- It helps to **identify abnormal activity spikes**, indicating moments when the overhead is the highest.



Analyzes to quantify the overhead

Time spent outside the vm

- It allows us to **measure how long the system takes to resume execution after an exit**.
- It helps detect if certain exits **cause abnormal latency**.



Analyzes to quantify the overhead

Correlation between exits and guest processes

- It allows us to **cross the origin of exits with the processes involved**.
- It helps detect **which applications generate the most costly exits**.

process	reason	count	percentage	total_delay_ms	mean_delay_ms	median_delay_ms	max_delay_ms
IDLE	EXIT_REASON_HLT	75	18.4729	16.0775	0.2144	0.114	2.9628
UNKNOWN	EXIT_REASON_EXTERNAL_INTERRUPT	15	3.6946	0.2719	0.0181	0.0072	0.1167
"swapper/1"	EXIT_REASON_MSR_WRITE	40	9.8522	0.0589	0.0015	0.0008	0.0079
"pool-tracker-mi"	EXIT_REASON_MSR_WRITE	15	3.6946	0.0493	0.0033	0.0021	0.009
"swapper/3"	EXIT_REASON_EPT_MISCONFIG	18	4.4335	0.0461	0.0026	0.0024	0.0037
UNKNOWN	EXIT_REASON_EPT_MISCONFIG	27	6.6502	0.0449	0.0017	0.0016	0.0023
"kworker/u10:1"	EXIT_REASON_MSR_WRITE	14	3.4483	0.0396	0.0028	0.0024	0.0062
"swapper/0"	EXIT_REASON_MSR_WRITE	13	3.202	0.0256	0.002	0.0019	0.0048
"gmain"	EXIT_REASON_MSR_WRITE	11	2.7094	0.0218	0.002	0.0023	0.0029
"dpdk-worker1"	EXIT_REASON_MSR_WRITE	12	2.9557	0.0203	0.0017	0.001	0.0038
UNKNOWN	EXIT_REASON_PAUSE_INSTRUCTION	13	3.202	0.0188	0.0014	0.0013	0.0039
UNKNOWN	EXIT_REASON_PREEMPTION_TIMER	14	3.4483	0.0167	0.0012	0.0011	0.0017
"swapper/3"	EXIT_REASON_MSR_WRITE	11	2.7094	0.0159	0.0014	0.0008	0.0076
UNKNOWN	EXIT_REASON_MSR_WRITE	15	3.6946	0.0128	0.0009	0.0008	0.002

Conclusion and in-progress

Summary

- The goal is to provide analysis that can help detect sources of overhead quickly
- For now 4 analyzes have been developed
- Still need to improve the correlation between the exits and the guest activity

Going further

- Integrate the analyzes into trace compass
- Test the analyzes on various type of workload (memory bound, cpu bound etc..)
- Integrate the analysis to provide context around exits (*overhead tax*)

Questions ?

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