High Frequency Performance Monitoring via Architectural Event Measurement

Chen Liu

cliu@clarkson.edu

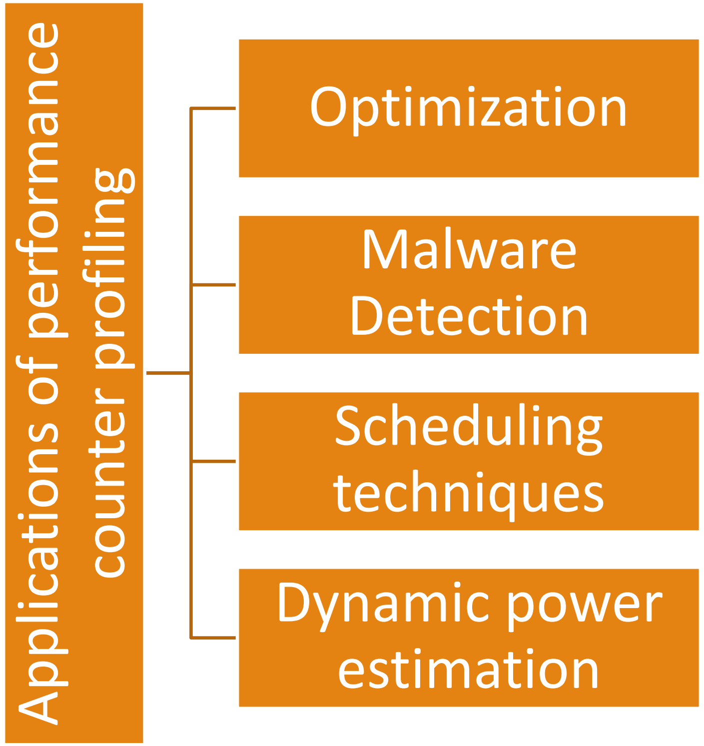
Clarkson University, Potsdam, New York, USA

SIGPLAN Benchmarking In the Data Center: Expanding to the Cloud (BID) Workshop April 2nd, 2022

Motivation

- There is significant demand to improve various performance metrics such as speed and energy efficiency within modern computing systems.

- The finer-grained performance details are commonly gathered using hardware performance counters that are built into modern processors.



Motivation

➢Many tools have been developed to provide a high-level API to control the low-level performance counters.

Tools

- Perf
- PAPI
- LiMiT

Limitations

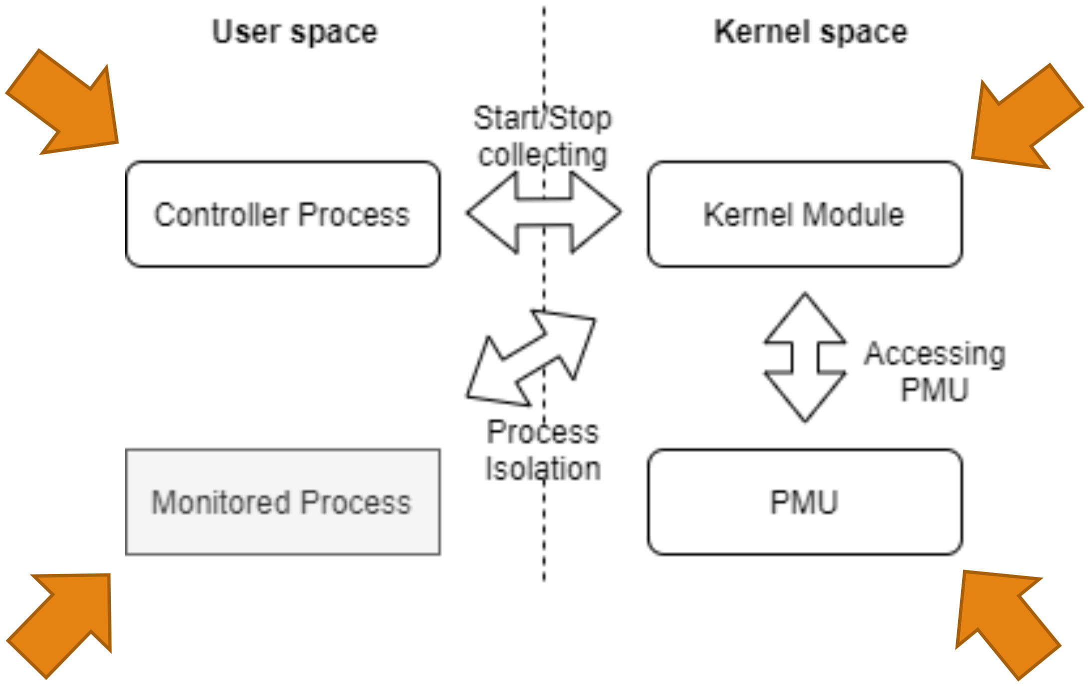
- Timer granularity
- Performance overhead
- Require Source code instrumentation
- Require kernel patch

Kernel - Lineage of Event Behavior (K-LEB)

A performance counter-based profiling tool that utilizes a kernel space collection system to produce ***precise, non-intrusive, low overhead, high periodicity*** performance counter data.

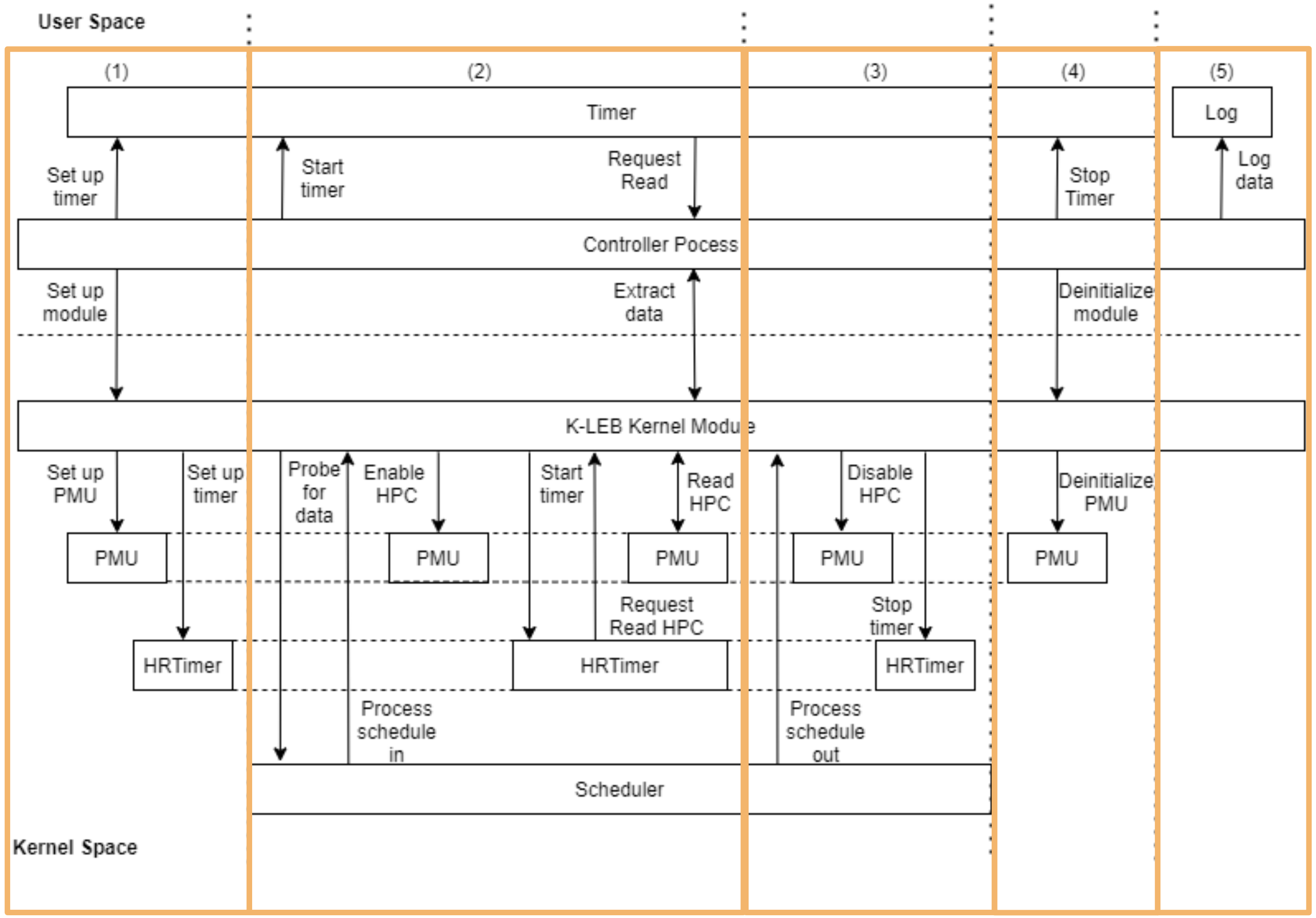
K-LEB System Model

- Controller process
  - Control the kernel module from user space.
- Kernel module
  - Access PMU to collect performance counter data.
- PMU
  - Special hardware registers use to monitor the hardware events.
- Monitored process
  - Process being monitored.



Process Flows

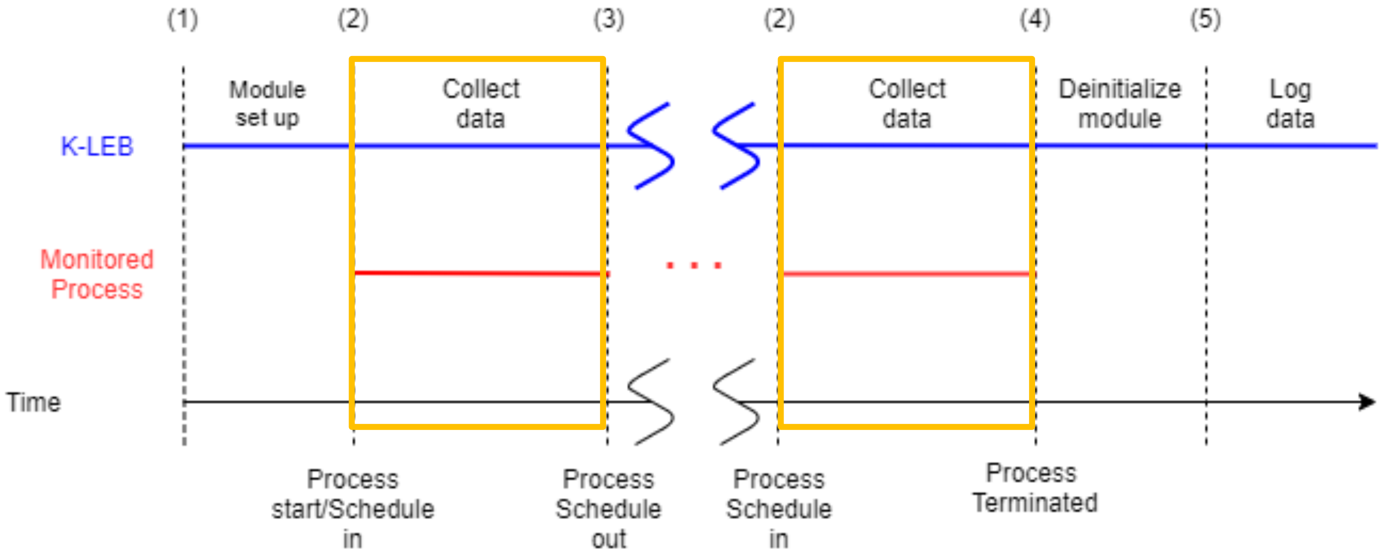
- 5 phases
  1) Module initialization
  2) Start monitoring
  3) Stop monitoring
  4) Module de-initialization
  5) Logging



Process Interaction

Interaction between K-LEB and the monitored process.

- 5 phases
  1) Module initialization
  2) Start monitoring
  3) Stop monitoring
  4) Module de-initialization
  5) Logging



Experiment setup

Intel Core i7-920 @ 2.67GHz Nehalem running Ubuntu 16.04 with Linux kernel version 4.13.0-15.

Intel Xeon Platinum 8259CL @ 2.50GHz Cascade Lake running Ubuntu 16.04 with Linux kernel version 4.4.0-1112-aws.

LINPACK
- Capture phase behavior
- Non-intrusive monitoring
- Performance overhead

Docker
- Workload characterization
- Non-intrusive monitoring

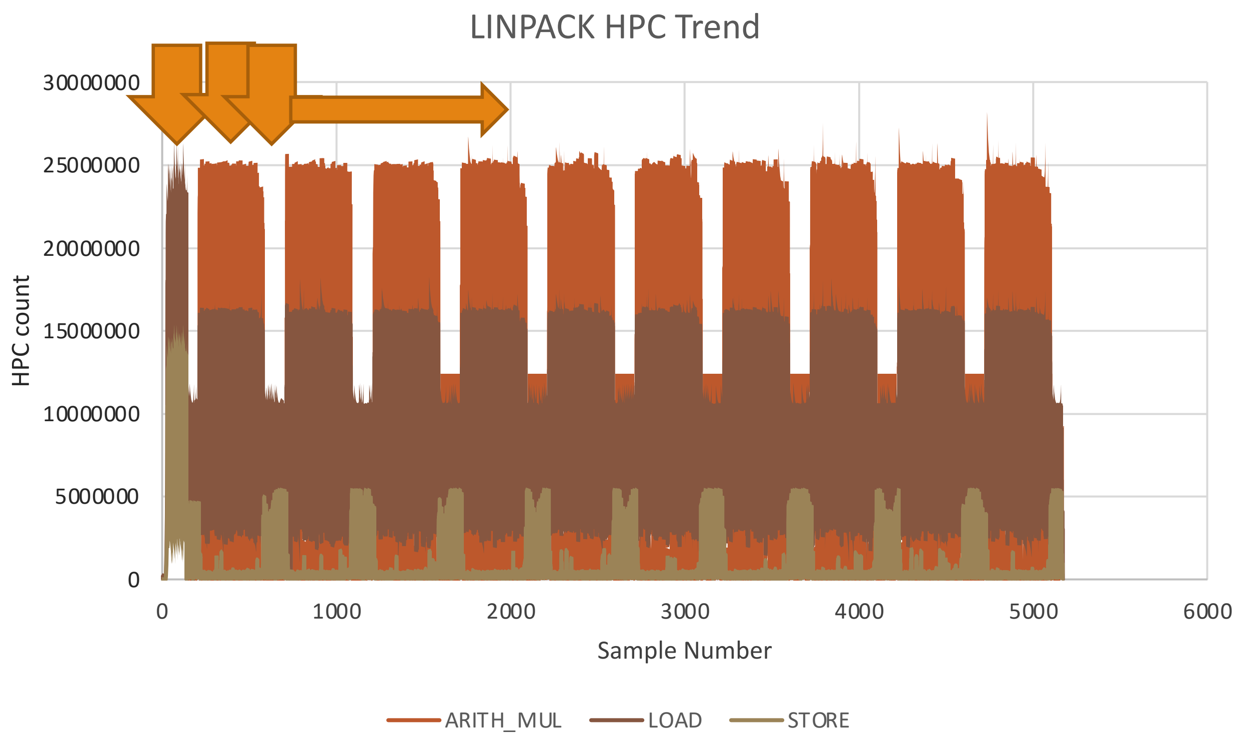
Meltdown
- Anomaly detection
- High timer granularity

Case study 1 LINPACK

- Capture phase behavior.

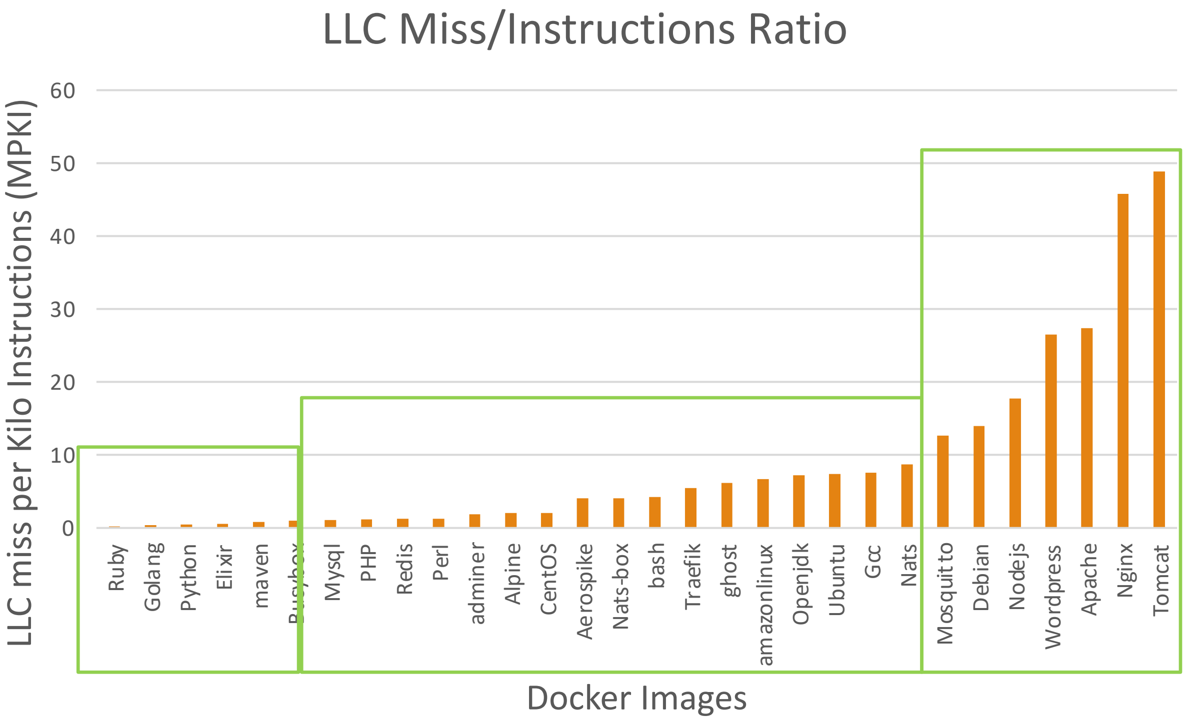
- K-LEB has a very small FLOPS loss of 0.64% in comparison with 7.08% from Perf.

- No source code require.



Case study 2 Docker

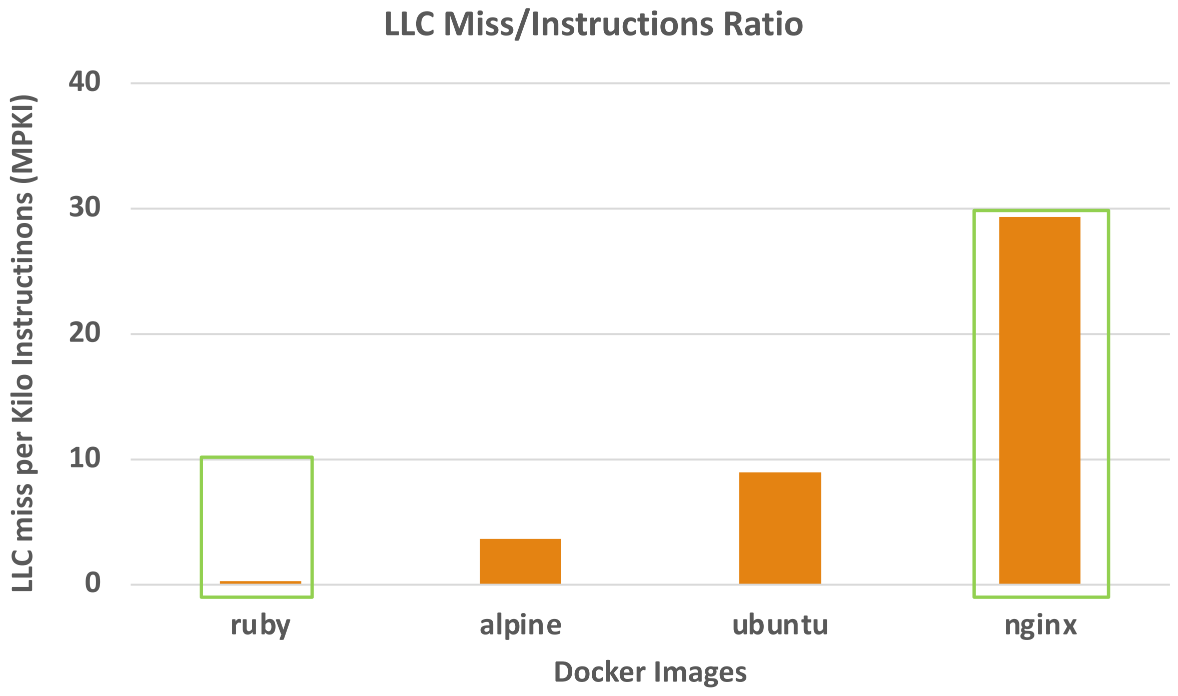
- ➢Workload characterization.
- ➢Computation/Memory intensive.
- ➢Non-intrusive to a running program.



Case study 2 Docker (continued)

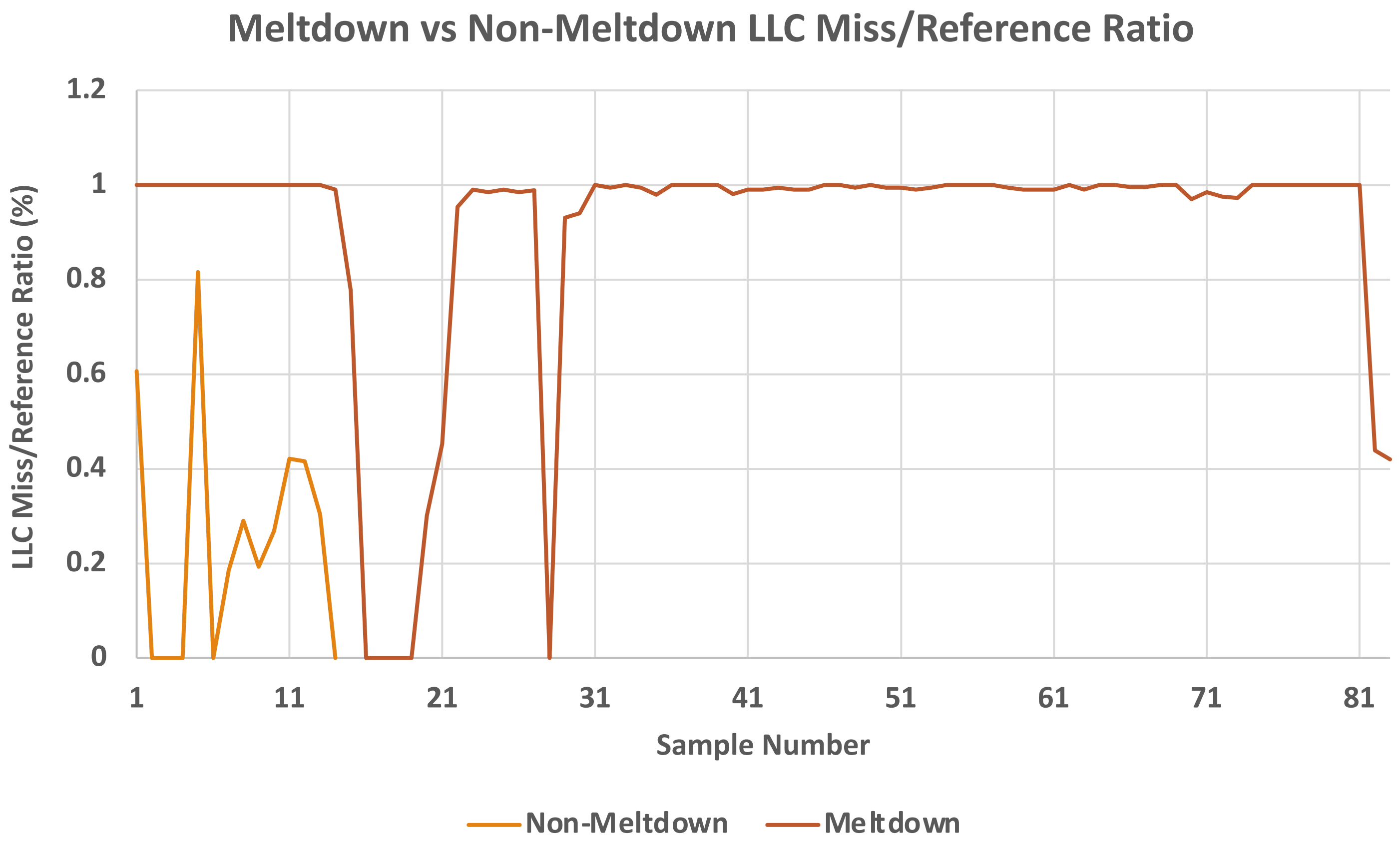
- Running on AWS machine.

- The programs still follow the same trend in terms of their MPKI from low to high.



Case study 3 Meltdown

- Anomaly detection.
- High frequency timer.
- Monitor program with short execution time.

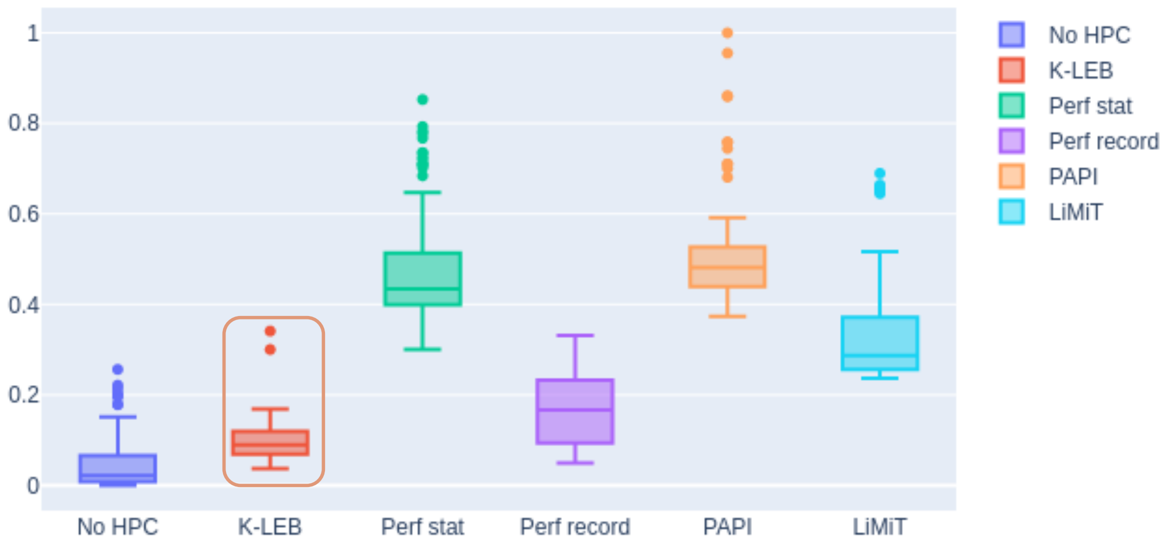


Performance overhead comparison

- Test on matrix multiplication program.
- Percentage performance overhead for each profiling tools.

Sample Rate	K-LEB	Perf stat	Perf record	Number of Samples	PAPI	LiMiT
10 ms	0.68	6.01	1.66	250	6.43	4.08
1 ms	0.8	N/A	2.15	2500	7.78	4.47
0.1 ms	1.48	N/A	6.55	25000	16.53	10.01

Normalized Execution Time



- ➢Test on matrix multiplication program.
- ➢K-LEB consistently has less spread in execution time across all comparable tool.

Hardware events count difference

- Percentage difference of hardware events count of K-LEB in comparison to other profiling tools.

	Branch	Load	Store	Instruction retired	Clock cycle
Perf stat	-7.95E-04	-6.29E-05	-3.90E-04	-5.23E-05	-0.30
Perf record	7.38E-03	-4.55E-03	-0.15	5.42E-03	0.03
PAPI	0.01	0.03	0.24	0.01	0.02

Conclusion

- In this work we introduce K-LEB, a kernel module-based approach for performance counter collection with following features.
  - Being non-intrusive to the program being monitored.
  - Can provide high frequency sampling rate up to 100μs, which is 100 finer granularity than current available tools.
  - Have very low overhead.
- This new approach allows users to better measure performance and behavioral characteristics of programs.
- As a result, many other subject areas that benefit from using performance data, such as program analysis, malware detection and scheduling techniques, could advance as well.

Acknowledgement

- Thanks Mr. Chutitep Woralert, Mr. James Bruska, and Dr. Lok Yan for working on the K-Leb project.

- https://github.com/camel-clarkson/k-leb

- https://camel.clarkson.edu/