On the Applicability of PEBS based Online Memory Access Tracking for Heterogeneous Memory Management at Scale

Aleix Roca Nonell, Balazs Gerofi ‡, Leonardo Bautista-Gomez,
Dominique Martinet †, Vicenç Beltran Querol, Yutaka Ishikawa ‡

Barcelona Supercomputing Center, Spain
†CEA, France
‡RIKEN Center for Computational Science, Japan

RIKEN

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

DE LA RECHERCHE À L'INDUSTRIE
cea

18/10/2018

Agenda

- Motivation
- Background
  - Lightweight Multi-Kernel OS
  - Processor/precise Event-Based Sampling (PEBS)
- Design
- Results
- Future Work
- Conclusions

Motivation

- Heterogeneous memories are here: HBM, MCDRAM, PCM, ReRAM, 3DXPoint, etc.
- Heterogeneous memory management alternatives:
  - Application level
  - Runtime level
  - Operating system level
- Operating system and/or runtime level
  - Application-transparent memory management eliminates complexity
  - Increased productivity/performance

- Need for low-cost real-time memory access tracking
- Is Processor Event based Sampling (PEBS) feasible when running on large-scale?
  - What are the trade-offs?

Objectives of this Paper

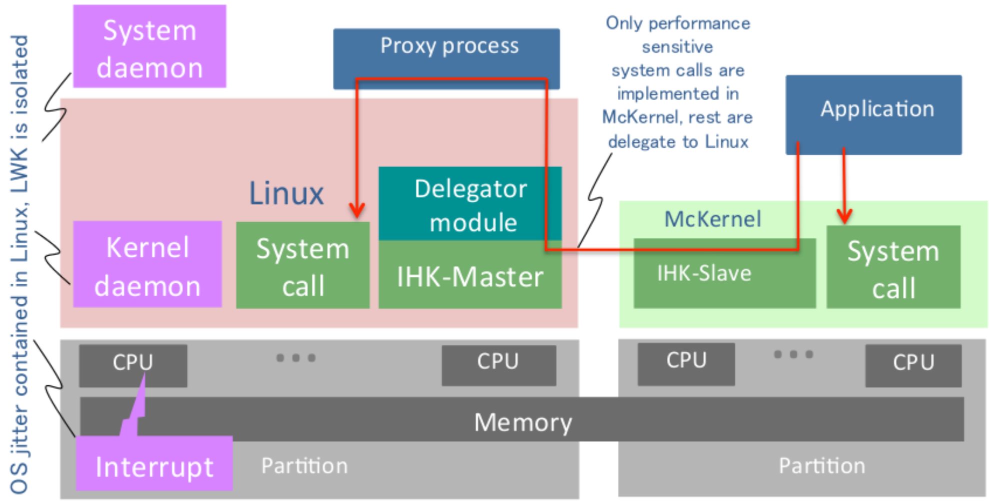
- Implement a custom PEBS driver in an LWK with the ability of fine-tuning its parameters
  - LWK provides a clean baseline to asses PEBS' overhead
  - Also due to Linux driver's limitations and instability
- Evaluate PEBS overhead on a number of real HPC applications running at large-scale
- Demonstrate captured memory access patterns as a function of different PEBS parameters

- **Analysis of PEBS overhead**
- We are not using the data to manage heterogeneous memory systems (yet)

Background: Lightweight Multi-Kernel OS

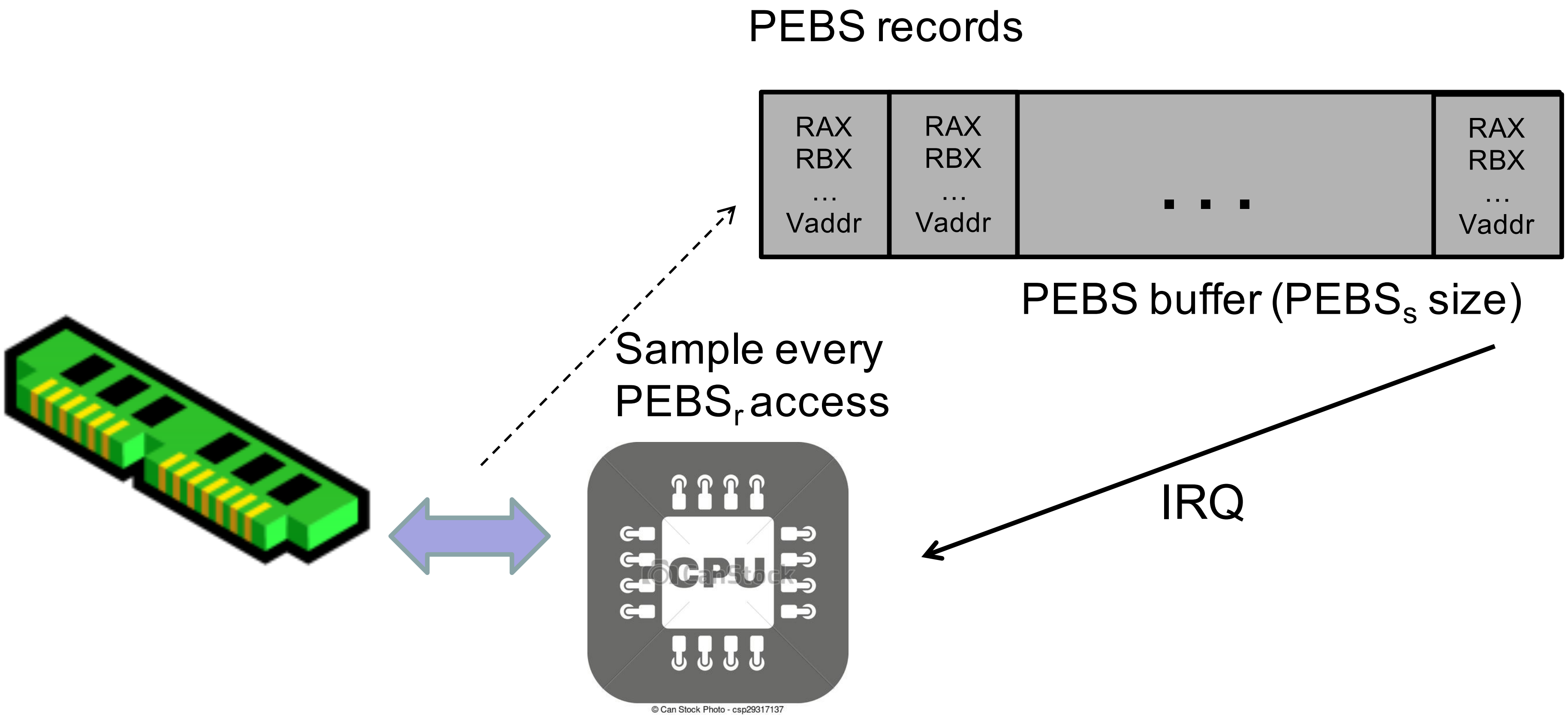
- IHK/McKernel:
  - Runs Linux and a lightweight kernel (i.e., McKernel) side-by-side on compute nodes
  - Interface for Heterogeneous Kernels (IHK) provides dynamic re-configurability of host resources
  - Management of LWK instances
  - McKernel is an LWK tailored for extreme-scale supercomputing (part of Post-K project)
  - Goal is to provide LWK scalability and full Linux/POSIX compatibility

- Merits for OS level memory management:
  - Simple LWK codebase allows rapid experimentation with specialized kernel features
  - Transparent usage of idle CPU cores for background data movement
  - Full control over HW resources
  - Ability to specialize drivers (e.g., PEBS)



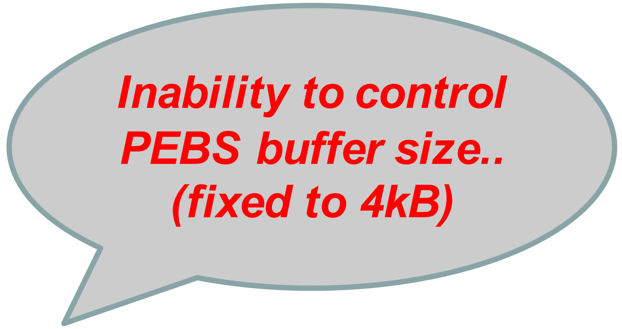
Background: Processor Event-Based Sampling (PEBS)

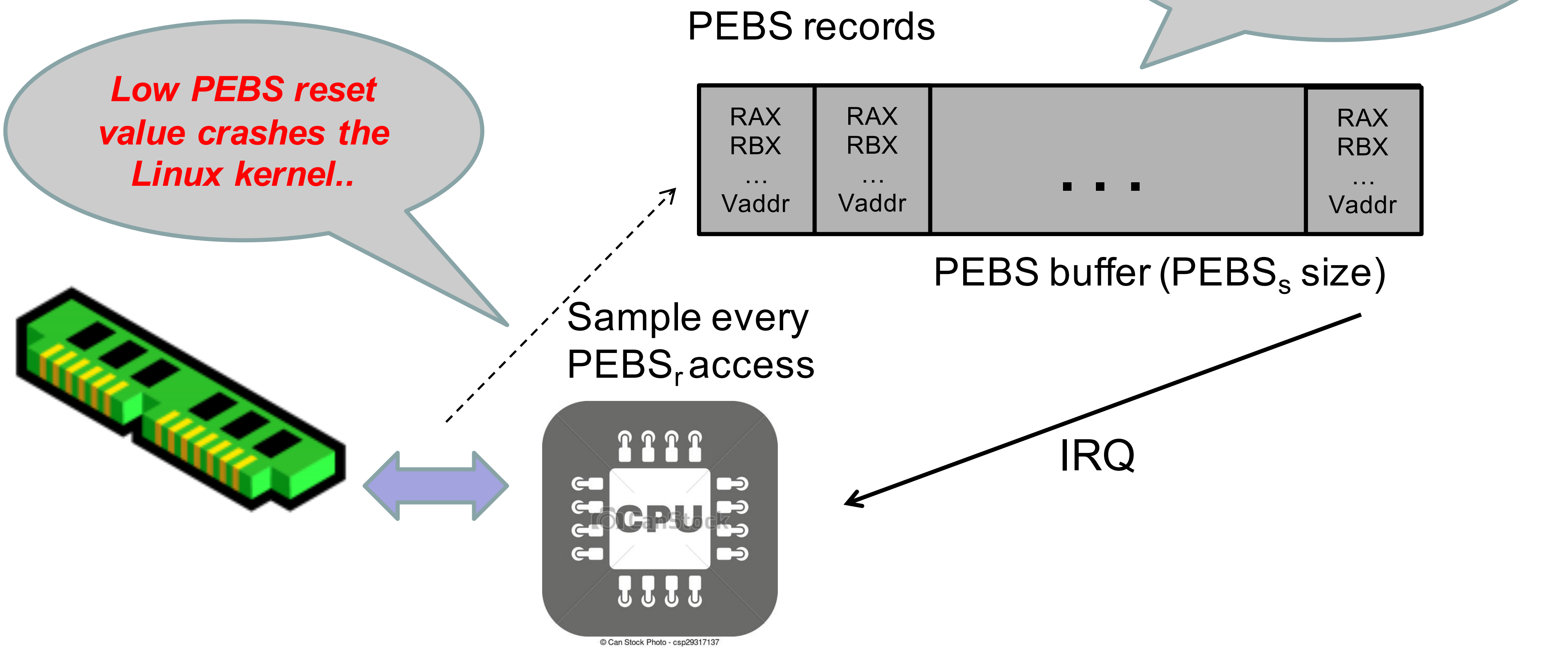
Extension to performance counters
PEBS reset: controls the sampling frequency
PEBS buffer size: indirectly controls IRQ frequency



PEBS Linux shortcomings

Extension to performance counters
PEBS reset: controls the sampling frequency
PEBS buffer size: indirectly controls IRQ frequency





PEBS Interrupt Rate Parameters

- Our focus is on PEBS interrupt rate
- Applications running at scale may suffer from noise introduced by asynchronous events such as IRQs

- PEBS' interference is affected by the following parameters:
  - Reset counter value: Event sample rate controls frequency on which PEBS records are written into the PEBS buffer
  - Buffer size: In-Memory buffer size (where PEBS records are stored) controls IRQ rate

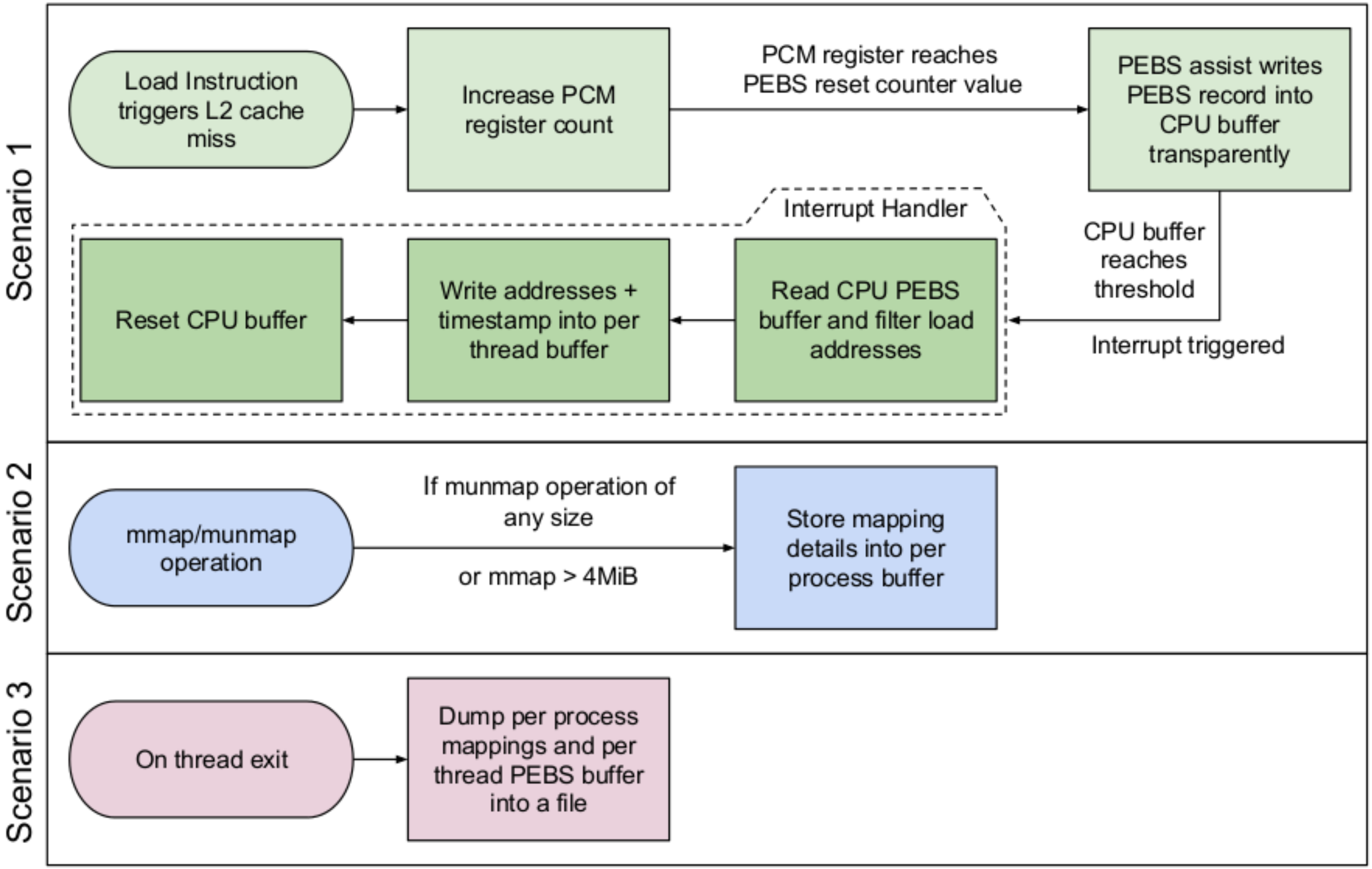
Design: Overview

McKernel provides a simple rapid-prototyping OS environment with low OS noise when compared to Linux

PEBS provides a configurable low-overhead mechanism to track memory accesses at runtime

McKernel + PEBS: groundwork for user-transparent heterogeneous memory management

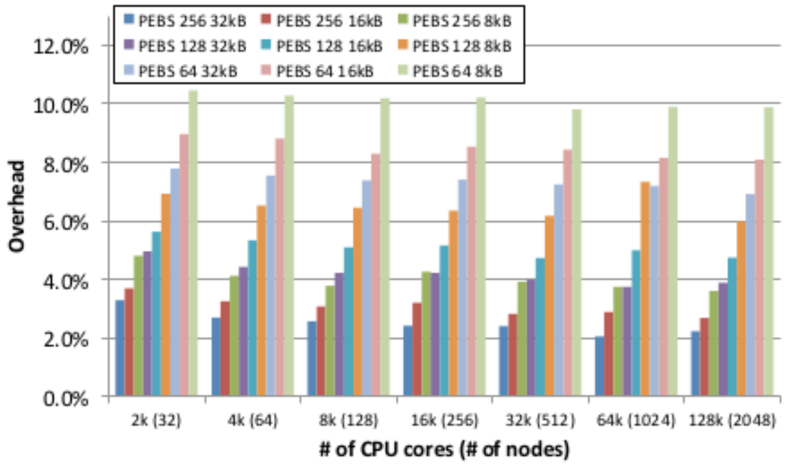
Design: McKernel + PEBS Architecture



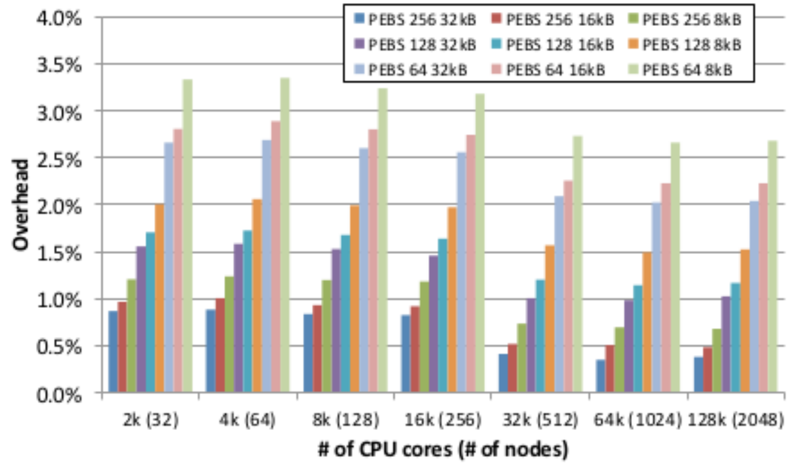
Evaluation: Oakforest-PACS

- 8k Intel Xeon Phi (Knights Landing) compute nodes
  - Intel OmniPath v1 interconnect
  - Peak performance: ~25 PF
- Intel Xeon Phi CPU 7250 model:
  - 68 CPU cores @ 1.40GHz
  - 4 HW thread / core
    - 272 logical OS CPUs altogether
  - 64 CPU cores used for McKernel, 4 for Linux
  - 16 GB MCDRAM high-bandwidth memory
    - Hot-pluggable in BIOS
  - 96 GB DRAM
  - **Quadrant flat mode**

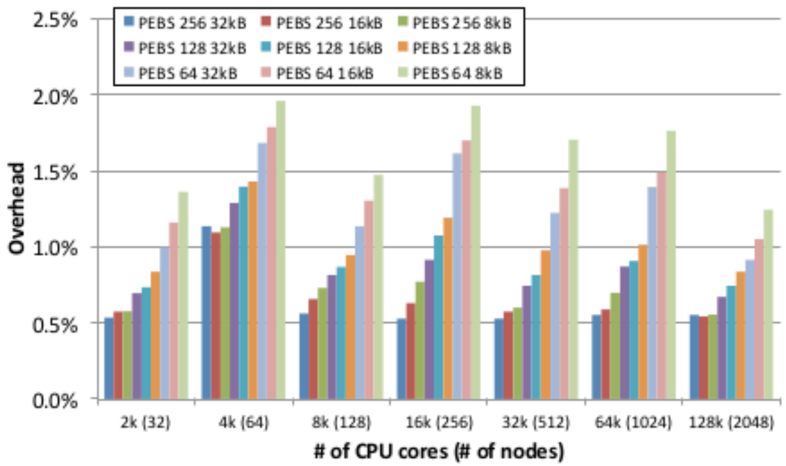
Results: PEBS overhead at scale @ Oakforest-PACS (OFP)



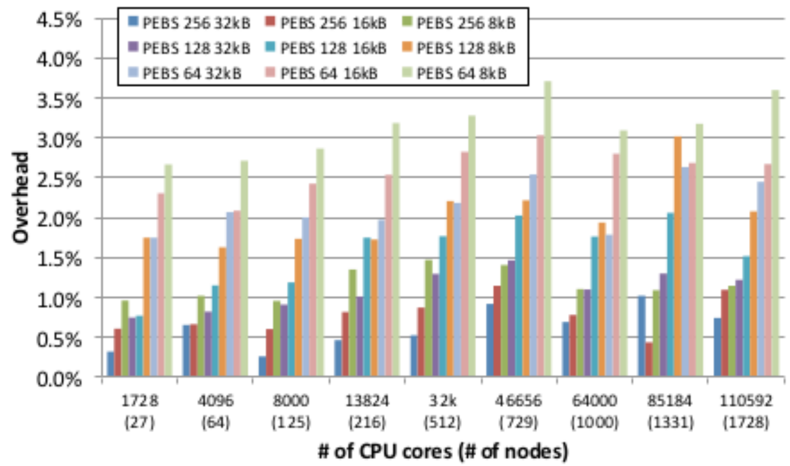
(a) GeoFEM (The University of Tokyo)



(b) HPCG (CORAL)

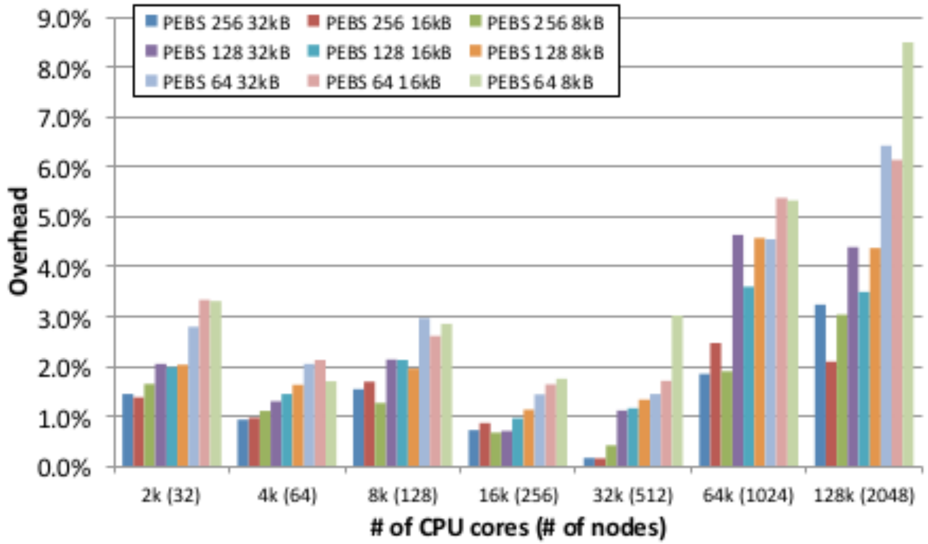


(c) LAMMPS (CORAL)

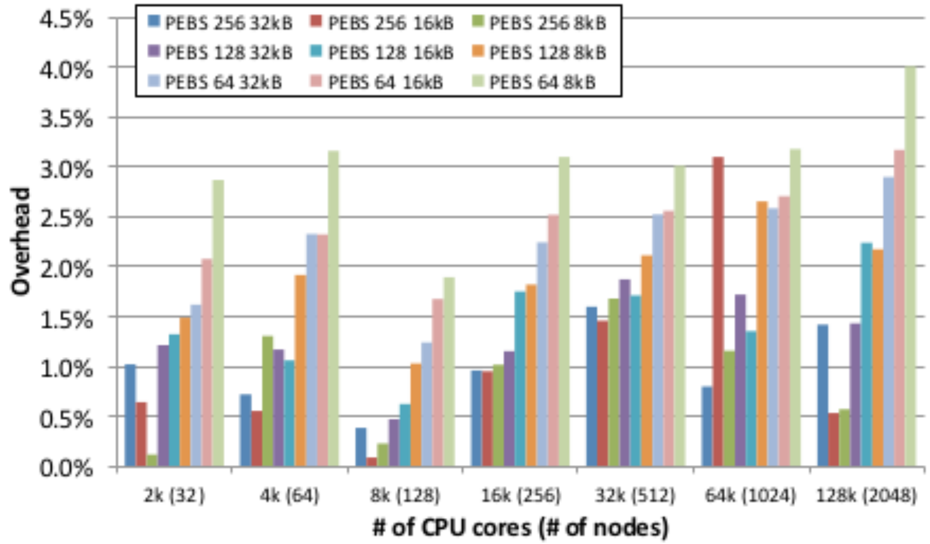


(d) Lulesh (CORAL)

Results: PEBS overhead at scale @ Oakforest-PACS (OFP)

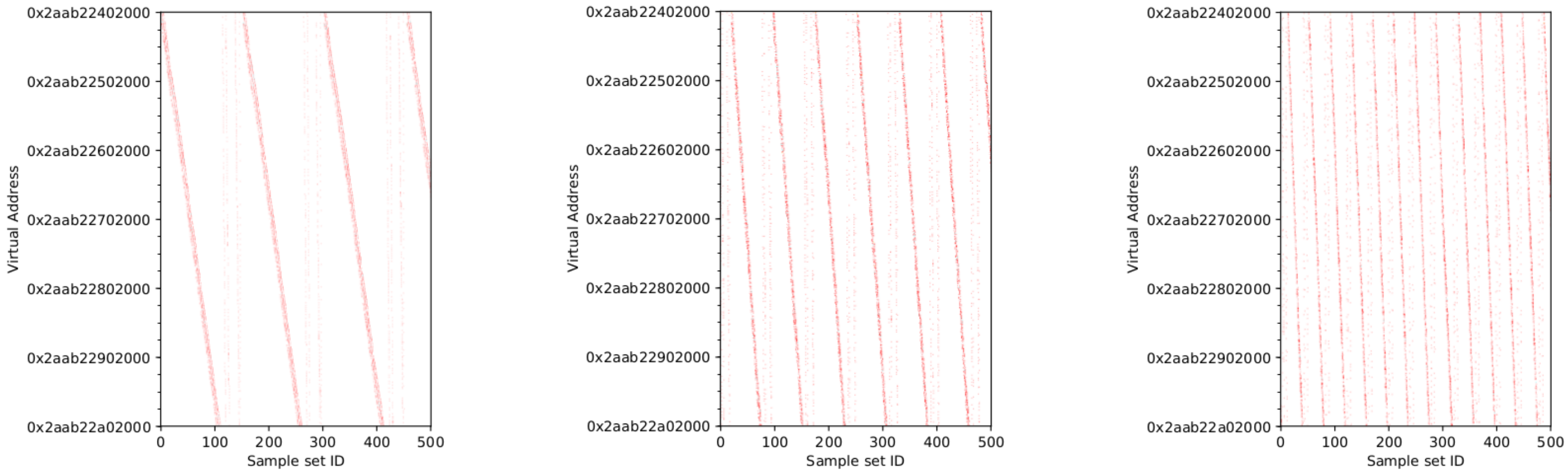


(e) MiniFE (CORAL)



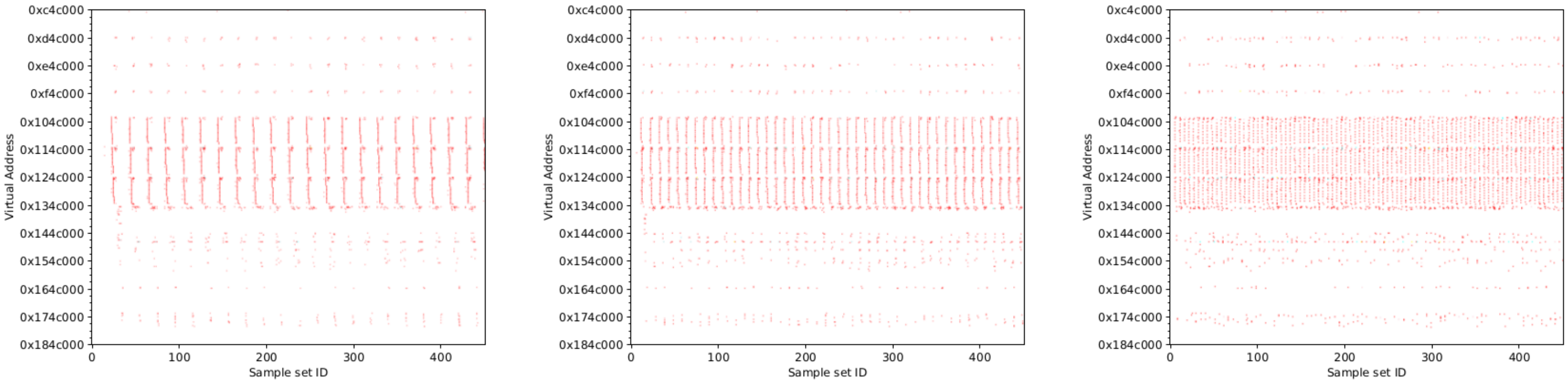
(f) AMG2013 (CORAL)

Results: Recorded access patterns for different PEBS reset values



(a) PEBS reset = 64 (b) PEBS reset = 128 (c) PEBS reset = 256

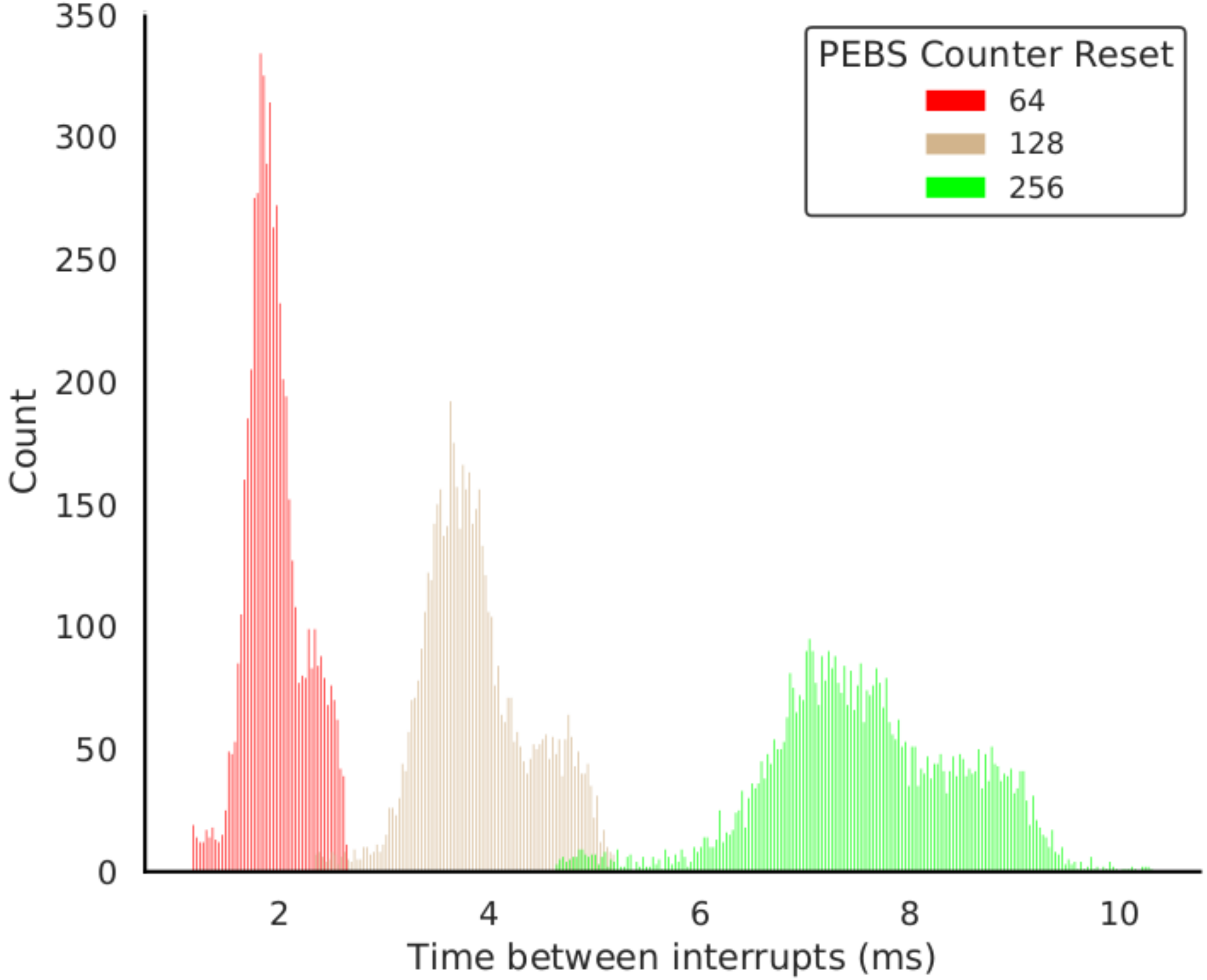
MiniFE access pattern with different PEBS reset values (8kB PEBS buffer)



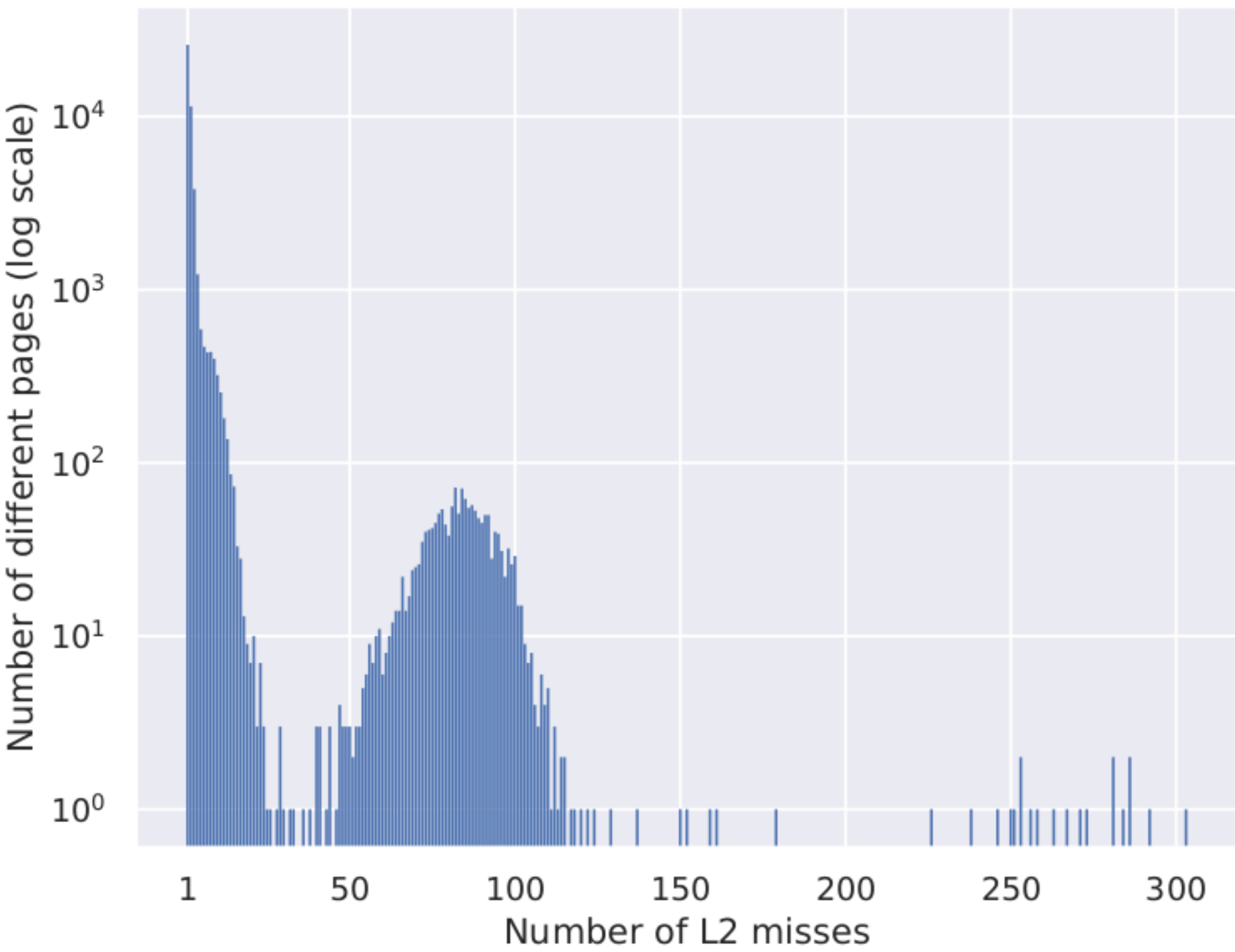
(a) PEBS reset = 64 (b) PEBS reset = 128 (c) PEBS reset = 256

Lulesh access pattern with different PEBS reset values (8kB PEBS buffer)

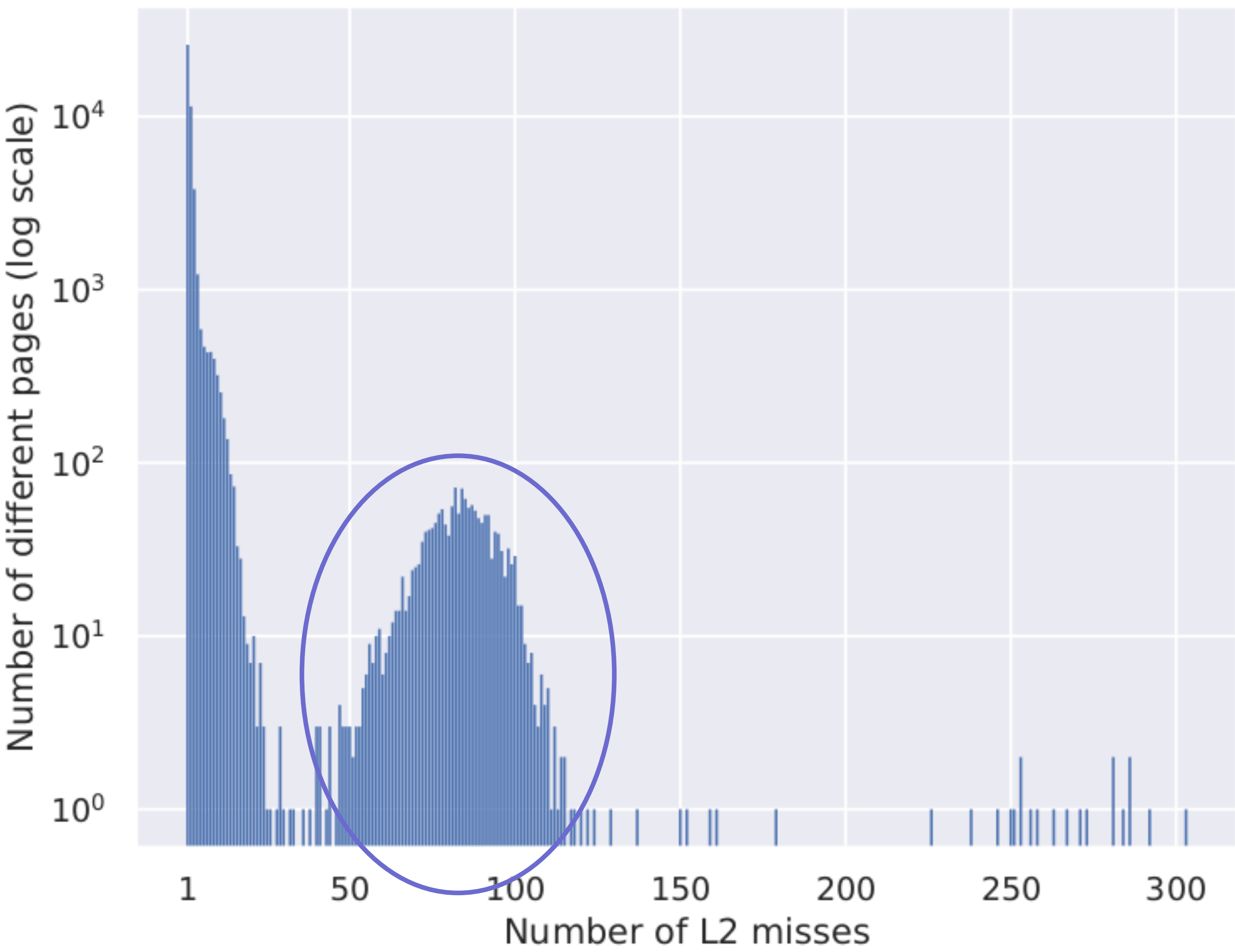
Results: Elapsed time between PEBS interrupts for MiniFE



Results: Access histogram per page for MiniFE



Results: Access histogram per page for MiniFE



Future Work

- Integration with un-core memory access traffic counters
- Study the possibility of a dedicated hardware thread to collect PEBS data instead of IRQs
- Analyse difference between McKernel and Linux PEBS driver
- Use profiled PEBS data for heterogeneous memory management
  - Machine learning for access prediction, memory placement

Conclusions

- Overheads range between 1% and 10.2% and that can be reduced to 4% by adjusting the recording parameters while still clearly capturing access patterns
- McKernel driver achieves more fine-grained sample rates than the Linux driver
- PEBS efficiency matches requirements for heterogeneous memory management

Thank you for your attention!
Questions?