Efficient Fault Tolerance using Intel MPX and TSX

Oleksii Oleksenko,
Dmitrii Kuvaiskii, Pramod Bhatotia, Christof Fetzer

TECHNISCHE UNIVERSITÄT DRESDEN

Pascal Felber

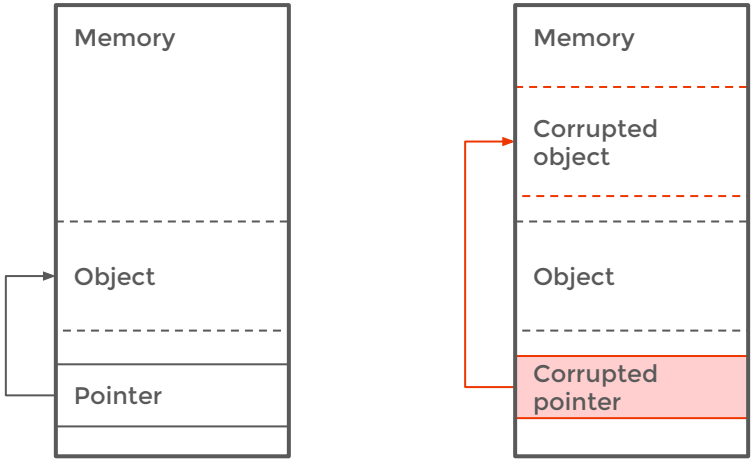
Université de Neuchâtel unine

Data corruption

- Performance-critical systems ➜ in a low-level language (C/C++)
- Low-level language ➜ no memory protection
  - Applications are more vulnerable to hardware faults

Data corruption

- Performance-critical systems ➜ in a low-level language (C/C++)
- Low-level language ➜ no memory protection
  - Applications are more vulnerable to hardware faults
- A pointer gets corrupted ➜ stays undetected



Overview

Problem:

- Existing solutions are expensive
  - They harden the entire program

Approach:

- Partial protection for efficient fault-tolerance
  - Protect only data pointers

Hypothesis

Leverage the new ISA extensions in modern CPUs for fault tolerance

Hypothesis

Leverage the new ISA extensions in modern CPUs for fault tolerance

Fault detection

Intel MPX

(Memory Protection Extension)

Fault recovery

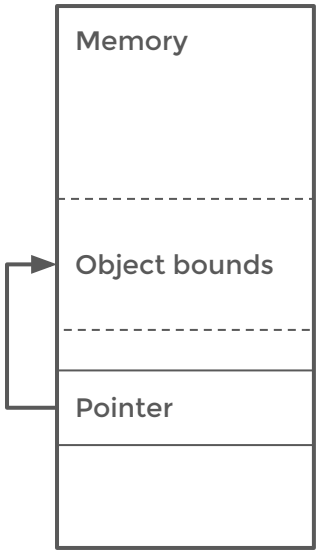
Intel TSX

(Transaction Sync Extensions)

Fault detection via Memory Protection Extension

Intel MPX: Bounds checking in the H/W

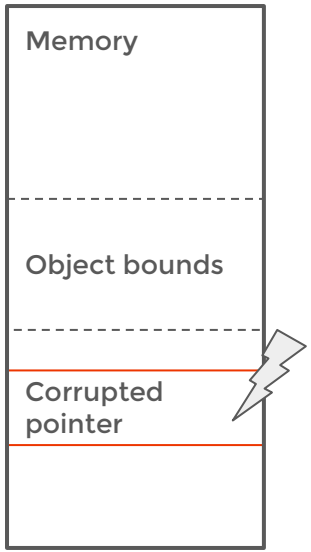
Insight: Pointer error will cause bound violation with high probability



Fault detection via Memory Protection Extension

Intel MPX: Bounds checking in the H/W

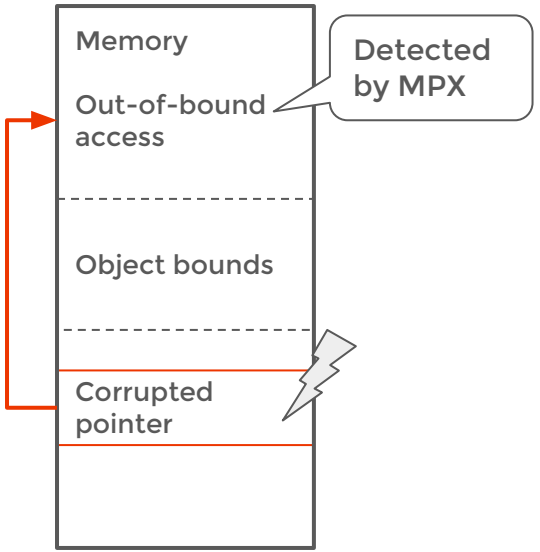
Insight: Pointer error will cause bound violation with high probability



Fault detection via Memory Protection Extension

Intel MPX: Bounds checking in the H/W

Insight: Pointer error will cause bound violation with high probability

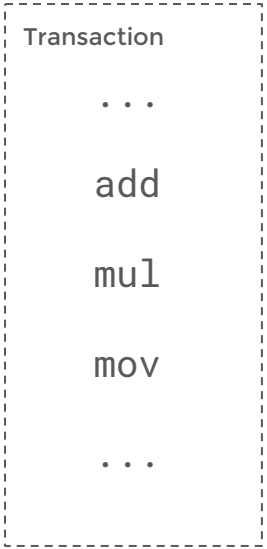


Fault recovery using transactions

Intel TSX: Transactions for optimistic concurrency control

Approach:

- Detect faults using MPX
- Use transactions for recovery

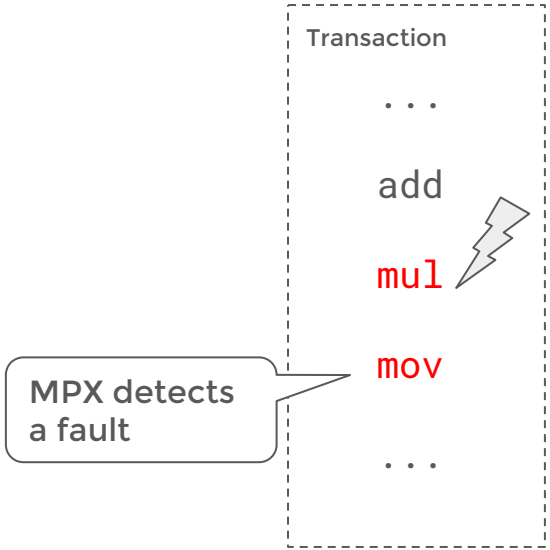


Fault recovery using transactions

Intel TSX: Transactions for optimistic concurrency control

Approach:

- Detect faults using MPX
- Use transactions for recovery

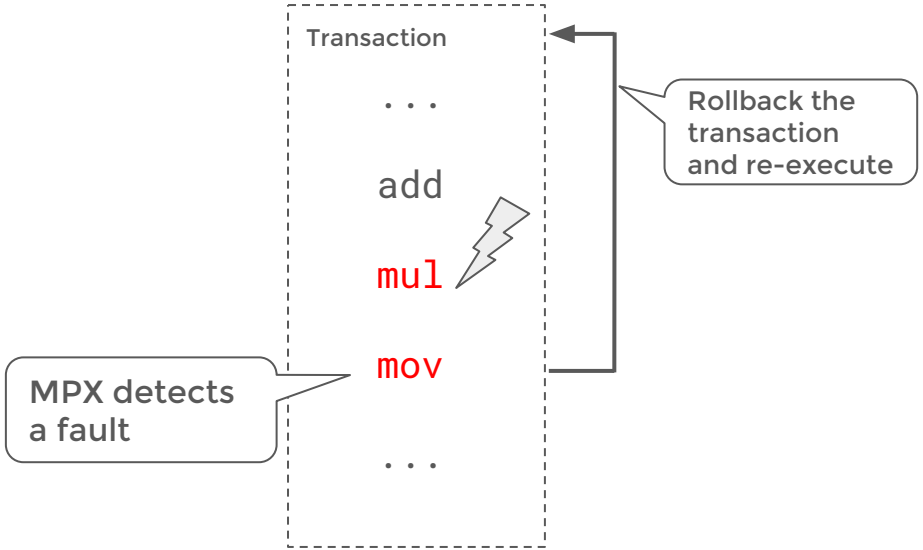


Fault recovery using transactions

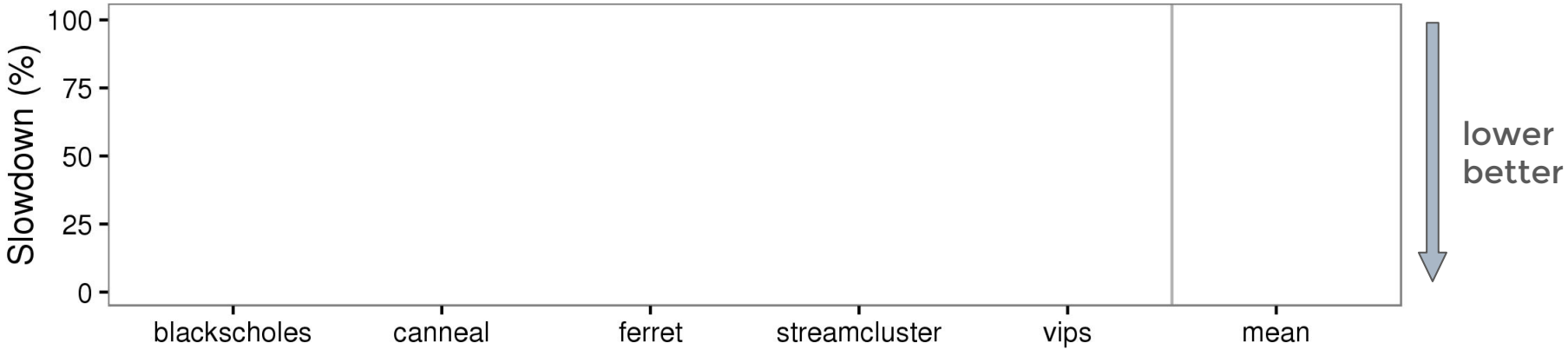
Intel TSX: Transactions for optimistic concurrency control

Approach:

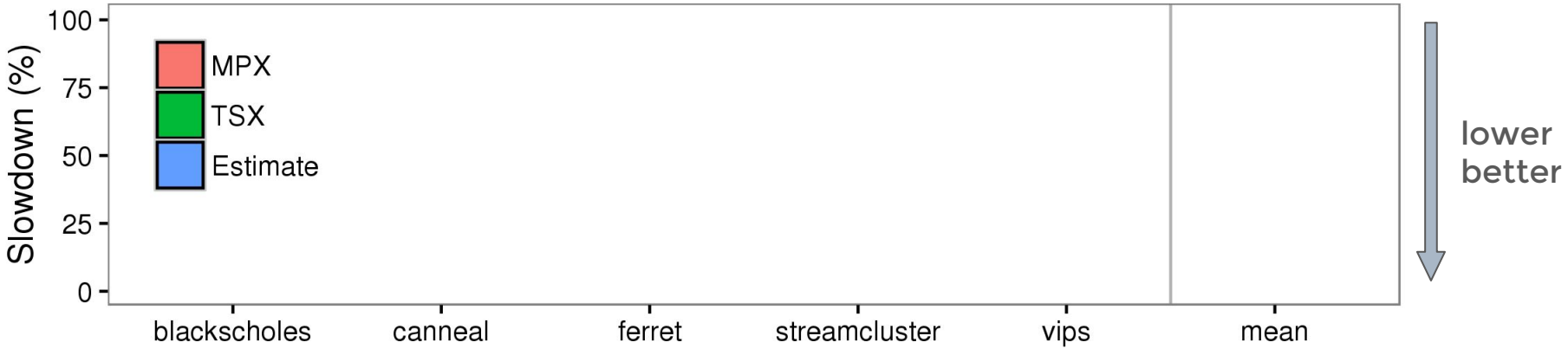
- Detect faults using MPX
- Use transactions for recovery



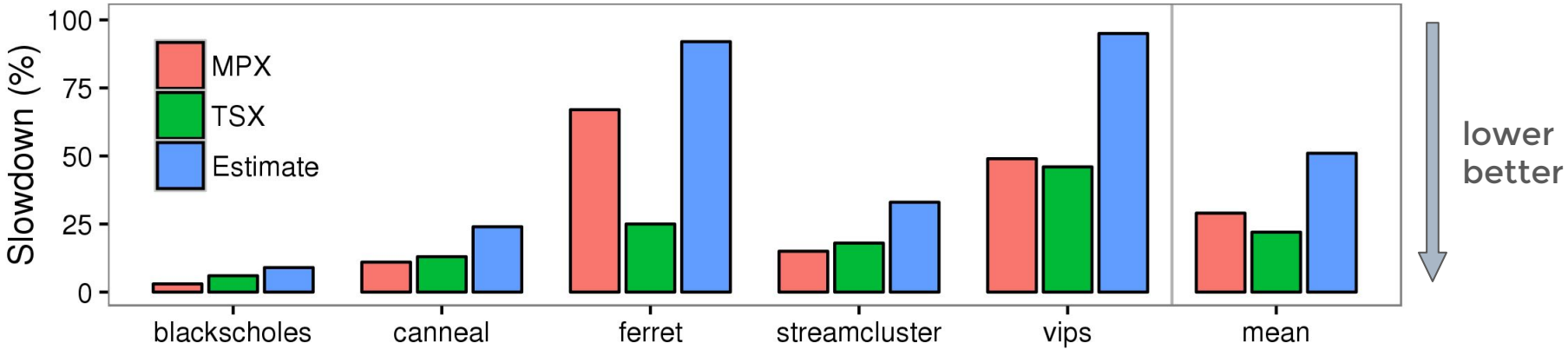
Performance overheads



Performance overheads



Performance overheads



~ 50% slowdown on average

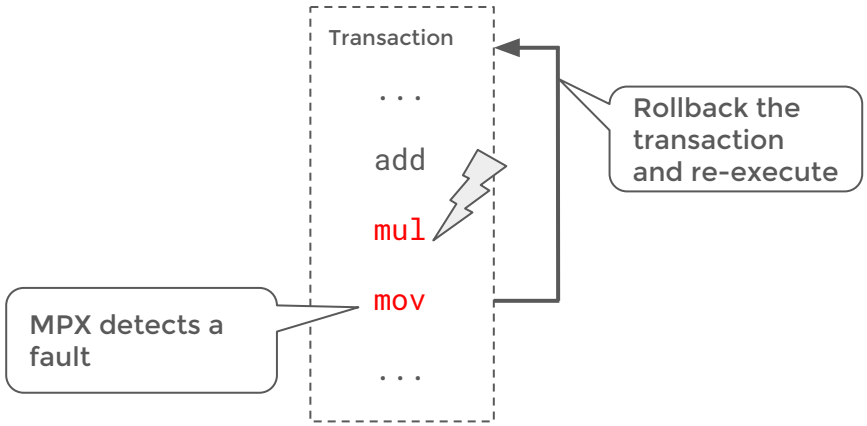
Summary

Leverage the new ISA extensions for fault tolerance:

- MPX: fault detection
- TSX: fault recovery

Improved efficiency:

- ~ 50% slowdown
  - State-of-the-art full hardening - 100%



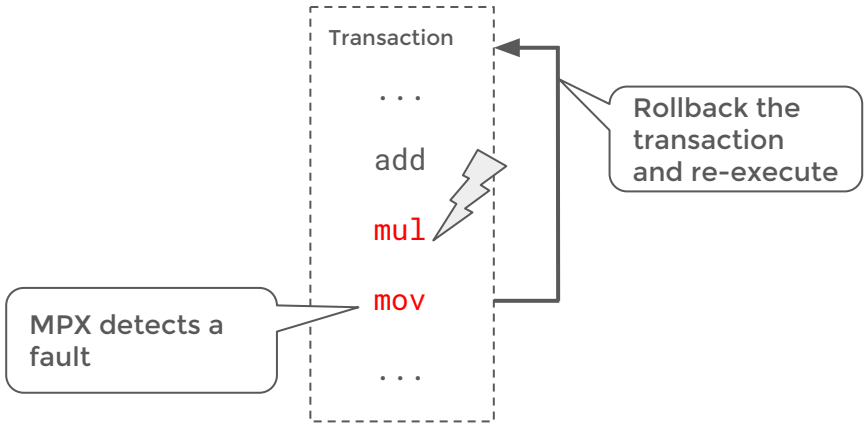
Summary

Leverage the new ISA extensions for fault tolerance:

- MPX: fault detection
- TSX: fault recovery

Improved efficiency:

- ~ 50% slowdown
  - State-of-the-art full hardening - 100%



Thanks!

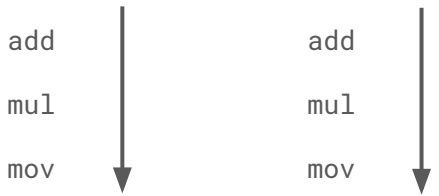
oleksii.oleksenko@tu-dresden.de

Backups

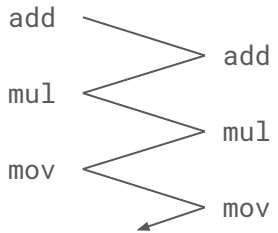
Generic Solutions

There are solutions for full-program hardening:

- Thread- and process-level redundancy
  - additional hardware
  - i.e., more cores used



- Duplicated execution
  - high performance overhead
  - x2 on average



Approach