EPVF: AN ENHANCED PROGRAM VULNERABILITY FACTOR METHODOLOGY FOR CROSS-LAYER RESILIENCE ANALYSIS

Bo Fang †, Qining Lu †, Karthik Pattabiraman †, Matei Ripeanu †, and Sudhanva Gurumurthi *
† The University of British Columbia, Canada
*Cloud Innovation Lab, IBM, USA
What are we facing?

- SoC soft error rate (SER) trends:

  **SoC FIT rate per node**

  - Bitcell SER
  - 200
  - 150
  - 100
  - 50
  - 0

  **MCU Avg/node**

  - SCU Avg/node

  
  Even though per memory bitcell SER sensitivity is decreasing, overall FIT per SoC is increasing.

  Source: iRoC
Why Software-based Fault Tolerance

- Hardware-based techniques

Software-based techniques: more cost-effective
Mitigating Silent Data Corruption (SDC): Key to Error Resilience
Error Resilience Estimation: Accuracy vs Cost

- Goal
- AVF/PVF
  - Conservative estimation of Error Resilience
- FI
  - High resource consumption, low predictive power

Accuracy vs Cost

[HPCA2010, MICRO2003]
Identifying SDC-causing Bits

- AVF/PVF: Identify Architecturally Correct Execution (ACE) Bits [MICRO03, HPCA10]

(enhanced)PVF: a methodology that distinguishes crash-causing bits from ACE bits
PVF Analysis [Sridharan, HPCA10’]

R1 = LD R2
R4 = ADD R1, R3
R5 = ADD R6*4, R7
ST R4, R5
R8 = LD R2

- ACE Bits = $\sum_{i=1}^{7} Bits\ in\ Ri$
- Total Bits = $\sum_{i=1}^{8} Bits\ in\ Ri$

- $PVF = \frac{ACE\ Bits}{Total\ Bits} = 88.9\%$
Our Approach: ePVF

- **Source of crashes**
  - Segmentation faults (99% of crashes are due to segfaults)

- **Direct crash-causing bits**
  - Crash model

- **Indirect crash-causing bits**
  - Propagation model
Overall methodology:

1. Obtaining Program Trace
2. PVF-Identify ACE bits
3. Crash Model
4. Propagation Model

- Identify bits that cause a program to make an invalid memory access and crash
- Identify bits on the backward slice of bits that directly cause crashes
Crash model

- Determining the bits that cause an out-of-bound memory access
- Applied on every memory instruction

R1 = LD R2
R4 = ADD R1, R3
R5 = ADD R6*4, R7
ST R4, R5
R8 = LD R2

R2 \in [addr_{min}, \, addr_{max}]
Propagation model

- Identifying all possible bits that can affect the bits identified by the crash model
  - R1 = LD R2
  - R4 = ADD R1, R3
  - R5 = ADD R6*4 + R7
  - ST R4, R5
  - R8 = LD R2

\[
\begin{align*}
\text{max}(R6) &= \frac{\text{max}(R5) - R7}{4} \\
\text{min}(R6) &= \frac{\text{min}(R5) - R7}{4}
\end{align*}
\]

\[
\begin{align*}
\text{max}(R7) &= \text{max}(R5) - R6*4 \\
\text{min}(R7) &= \text{min}(R5) - R6*4
\end{align*}
\]
Overall ePVF methodology

- Obtaining Program Trace
- PVF-Identify ACE bits
- Crash Model
- Propagation Model

ePVF Bits that potentially lead to SDCs
Experimental setup

- Scientific benchmarks
  - 8 from Rodinia [IISWC 09]
  - Matrix Multiplication
  - LULESH: DOE proxy app [IPDPS 2013]

- Fault Model

- LLFI [DSN 14]
  - 3,000 runs per benchmark
Evaluation

- RQ1: Accuracy of the models
- RQ2: Effectiveness of the ePVF methodology
- RQ3: Performance
RQ1: Accuracy of the models

- **Recall**

- **Precision**

Our models achieve average 89% recall and 92% precision.
RQ1. Accuracy of the Models

On average, 90% of the time the ePVF methodology is accurate to identify crash-causing bits

On average, 90% of the time the ePVF methodology is accurate to identify crash-causing bits
RQ2: Effectiveness of the ePVF

- SDC estimate using PVF analysis, ePVF analysis and Fault Injection

![Diagram showing ePVF significantly tightens the upper bound of estimated SDCs by 61% on average]
ePVF-informed Duplication

- Rank instructions based on their ePVF value

  - ePVF value per instruction = \( \frac{ACE \text{ bits} - \text{Crash-causing bits}}{ACE \text{ bits}} \)
  - Higher the ePVF value, Higher chance to lead to SDCs

- Duplication highly-ranked ePVF instructions

- 30% more SDC coverage than hot-path duplication for the same performance overhead
RQ3: Performance

- Modeling time ranges from 30s (lavaMD) to ~ 4 hours (pathfinder).
  - Depending on the size of the DDG, hence the number of dynamic instructions

- Optimization (Sampling and Extrapolation)
  - Intuition – scientific applications usually have repetitive behaviors.

Extrapolated ePVF values based on 10% of the graph, and showing less than 1% difference on average
Conclusion

- ePVF removes the crash-causing bits from PVF to get a more accurate estimate of SDC rate.
  - A crash model that predicts direct crash-causing bits
  - A propagation model that identifies bit that lead to direct crash-causing bits
  - Implementation with LLVM compiler
  - Drive selective protection of SDC-causing instructions

Email: bof@ece.ubc.ca
Code: https://github.com/flyree/enhancedPVF