BonVoision: Leveraging Spatial Data Smoothness for Recovery from Memory Soft Errors

Bo Fang, Hassan Halawa, Karthik Pattabiraman, Matei Ripeanu, and Sriram Krishnamoorthy
Motivation

Transient hardware faults

- LetGo [HPDC17’]
- BonVoision
Memory errors affecting HPC systems

- 260 TB DRAM
- One DRAM error every 11 hour

[Sridharan et al. SC 2013]

- 1.47 PB RAM
- 160/h memory errors across half of nodes

[Martino et al. DSN 2014]
Handling memory DUEs in software

Application
OS
MCE
CPU
DUE
ECC Memory
C/R
SIG

Diagram showing the timing and handling of memory DUEs with symbols for t_c, t_i, t_{i+1}, t_{i+2}, and t_R.
Roll-forward C/R: mitigate overhead

Benefits:
- longer MTBF, longer ckpt interval
- no restart overhead

Roll-forward recovery for memory DUEs

1. Effectiveness
2. Efficiency
Our approach: BonVoision
Spatial data smoothness

Assumptions:

- Modeled physical space may exhibit inherent continuities

- Logical space mapping to physical space preserves data locality

Smoothness is observed across scientific applications
Exploiting the spatial data smoothness

```c
for (int i = 0; i < n; i++)
{
    double mass = a[i].m;
    // do some computation
    // ... 
}
```

Goal: use values of nearby elements to repair

Available info:
- memory address
- size
- data type

<table>
<thead>
<tr>
<th>Stride</th>
<th>size</th>
<th>Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bytes (Mass)</td>
<td>8 bytes (Velocity)</td>
<td>8 bytes (Energy)</td>
</tr>
<tr>
<td>8 bytes (Velocity)</td>
<td>8 bytes (Energy)</td>
<td>8 bytes (Mass)</td>
</tr>
<tr>
<td>8 bytes (Motion)</td>
<td>8 bytes (Velocity)</td>
<td>8 bytes (Energy)</td>
</tr>
<tr>
<td>8 bytes (Velocity)</td>
<td>8 bytes (Energy)</td>
<td>8 bytes (Mass)</td>
</tr>
<tr>
<td>8 bytes (Motion)</td>
<td>8 bytes (Velocity)</td>
<td>8 bytes (Energy)</td>
</tr>
<tr>
<td>8 bytes (Velocity)</td>
<td>8 bytes (Energy)</td>
<td>8 bytes (Mass)</td>
</tr>
</tbody>
</table>
BonVoision: stride speculation

Design space

Function annotation

Trace file flushing

Our approach

Stride speculation
Binary analysis
Triggered only by DUE
BonVoision: stride speculation

Key insight
- Different elements are accessed with the same addressing mode
- Modification on the some registers infers the stride

```assembly
4004c8: addsd (%rax,%rdx,1),%xmm0
4004cd: add $0x10,%rdx
4004d1: cmp $0x640,%rdx
4004d8: jne 4004c8 <main+0x38>
```
Evaluation questions

- Q1: How effective is BonVoision to recover applications from memory DUEs?
  - Can machine learning help improve BonVoision’s effectiveness?
- Q2: How efficient is BonVoision?
- Q3: What is the impact of BonVoision on the C/R?

Evaluation methodology

- **Fault injection: Q1, Q2**
- Simulations: Q3
Fault injection methodology

### Outcome Description

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crash</td>
<td>Exception</td>
</tr>
<tr>
<td>SDC</td>
<td>Undetected incorrect results</td>
</tr>
<tr>
<td>Benign</td>
<td>Correct results</td>
</tr>
<tr>
<td>Detected</td>
<td>Detected incorrect results</td>
</tr>
<tr>
<td>Hang</td>
<td>Time out</td>
</tr>
</tbody>
</table>

- Profile to count memory load insts
- Randomly pick a load inst
  - Observe outcome
  - Normal execution
- Overwrite the memory with different repair strategies

Key Insights
BonVoision:
- highest rate of benign outcomes
- 0.1% to 2% increase in SDC rates for four benchmarks
BonVoision-E: optimization with ML

Problem: SDC and detected outcomes
  ▪ Detected cases cause applications to waste work

Goal: predict the outcome of a repair online
  ▪ If SDC or detected, stop
  ▪ If benign, continue

Methodology: supervised machine learning
Online Classifier Results

BonVoision-E converts on average 97% of not benign cases to crashes
Summary

Spatial data smoothness for recovery memory DUEs

A light-weight, automatic, and effective optimization for C/R

Trade off confidence in results for efficiency

Bo Fang’s website: https://flyree.github.io/
Netsyslab: http://netsyslab.ece.ubc.ca
DependableSystemsLab: https://github.com/DependableSystemsLab