ReplayCache: Enabling Caches for Energy Harvesting Systems

Jianping Zeng\textsuperscript{1}, Jongouk Choi\textsuperscript{1}, Xinwei Fu\textsuperscript{2},
Ajay Paddayuru Shreepathi\textsuperscript{3}, Dongyoon Lee\textsuperscript{3},
Changwoo Min\textsuperscript{2}, Changhee Jung\textsuperscript{1}

\textsuperscript{1}Purdue University
\textsuperscript{2}Virginia Tech
\textsuperscript{3}Stony Brook University
Battery is Not the Way to Go!

- Batteries are bulky
- They must be replaced

Annual Production of IoT devices

Source: SoftBank and ARM estimates
Energy Harvesting Systems (EHS)
Unreliability of Ambient Energy Sources

(a). Time (s) (Sample time: 0.33us)

(b). Time (s) (Sample time: 0.33us)

[Ma,HPCA'2015]
Problem of Frequently Data Loss

Timeline of program execution

Stagnation

\[ \infty \]

: Power failure on which all volatile registers lose their data
NVP: Just-in-time Register Checkpointing

➢ Register access is not delayed because of no NVFF access during program execution.
➢ No-rollback: program resumes at the exactly failure point.

Voltage monitor
Backup/recovery controller
Volatile register
Pipeline
NVM
Vbk
Energy
Voff
NVP
Voltage monitor

Harvested energy

Capacitor charge
Vbk
Voff
NVM
[Ma,HPCA'2015]

Register access is not delayed because of no NVFF access during program execution.
No-rollback: program resumes at the exactly failure point.

[Ma,HPCA'2015]

54th IEEE/ACM International Symposium on Microarchitecture
NVP Has No Cache Due to Its Crash Consistency Issue

* We can’t restart program from the exactly failure point as NVP does
NVSRAM: The State-of-the-Art and its Limitations

- Secured energy is used only for backup; less progress
- Slow booting for securing the higher voltage
- High manufacturing complexity

[Chui, VLSI'2010]
ReplayCache: A Pure Software Solution to Enabling Performant Volatile Cache for EHS

* Software undo/redo logging slow down 1.6-5x
Reason for Crash Inconsistency of Volatile Cache

Crash inconsistency caused by stores left behind power failure

r4 = Load [r3]

Power interrupted

* r2, r3 are restored from NVFF

New data

Old data

JIT Backup Recovery

NVM

RF

NVFF

Store r2, [r3]

...
ReplayCache Solution: Replaying Unpersisted Stores

* Recovery status after power failure happens

Power interrupted

Store r2, [r3]

... r4 = Load [r3]
Store Integrity: A Property to Ensure Correct Replaying

- **Store integrity**: register operands of stores must not be overwritten by following definitions.

(a). Code of no store integrity
- Store r0, [r1]
- \( r1 = \text{Load } [r2] \)

(b). Store integrity code by renaming
- Store r0, [r1]
- \( r3 = \text{Load } [r2] \)

Write-after-read (WAR) dep.
Challenge of Guaranteeing Store Integrity

- **Limited** number of registers prevents store integrity

Assume only 4 registers

Write-after-read (WAR) dep
Region-Level Store Integrity and Implications

Region level register reuse

Region level store persistence

➢ Safe to overwrite store register across regions
Region-Level Store Persistence

* CLWB: asynchronously write back store to memory.
Recovery Protocol

➢ Stores left behind failure are the root of cause

➢ Replaying them in the wake of failure with store integrity

➢ Due to limited registers, region-level store integrity is introduced, which in turn requires region-level store persistence to allow each region to use all registers.

➢ Recovery protocol
Just Restart a Power-interrupted Region?

- Re-executing the region from its beginning **fails** to recover program status

```
... Store r0, [r1]
... Store r2, [r3]
  r0 = Load [r2]
  r3 = r0 << 2
  ...  
```

Power interrupted
Region-Level Recovery Protocol

* Program Status and NVM status during recovery

Recovery Code

Store r2, [r3]
Store r2, [r3+4]

access NVM registers r2 and r3 in recovery code

ST1

... Store r0, [r1]
... r2 = r0 << 2
... Store r2, [r3]
... Store r2, [r3+4]
... r0 = Load [r2]
... NVM

Power interrupted

Program Status and NVM status during recovery

ST1
Challenge to Generate Store-Integrity Regions

Region partitioning

Register preservation (register renaming)

Original program

ReplayCache Compiler

Region partitioned program
No Store Integrity with Existing Register Allocation

- Assume only 3 registers
- Disjoint live ranges can share the same physical register (x,z)

Register Assignment:

- x ----> r0
- y ----> r1
- z ----> r0
- p ----> r1
- q ----> r1
Register-Renaming-Aware Region Partitioning

- Store operands must **not share** same registers with following definitions
- Assume 3 registers

### Pre-RA
- `call, ...
- `x = ...
- `y = ...
- `st y, ...
- `st x, ...
- ...
- `z = ...
- `p = ...
- `q = z<<2

### Post-RA
- `call, ...
- `r0 = ...
- `r1 = ...
- `st r1, ...
- `st r0, ...
- ...
- `r2 = ...
- `r1 = ...
- `q ----> r1
- `p ----> r1
- `q ----> r1

### Extended live ranges
- `x
- `y
- `z

### 4 overlaps
- `z
- `p
- `q

### Register Assignment
- `x ----> r0
- `y ----> r1
- `z ----> r2
- `p ----> r1
- `q ----> r1
- `r1 = r2<<2
Spilling-Store-Registers Preservation

```
call, ...
 r0 = ...
 r1 = ...
 st r1,...
 st r0,...
 ...
 r2 = ...
 ...
 r1 = ...
 st r1...
 r1 = r2<<2
```

Register Assignment

- x ----> r0
- y ----> r1
- z ----> r2
- p ----> r1
- q ----> r1
Gem5-based **NVPSim** modeling a single core in-order ARMv7 processor with 8kB 2-way set-associative L1 I/D cache, and 16MB ReRAM as main memory.

- LLVM-based region formation.
- Mediabench + Mibench.
Speedup over No-cache Baseline with Real Power Trace

- NVCache: 3.3
- NVSRAM: 11.3
- WT-VCache: 4.2
- ReplayCache: 8.9
ILP Efficiency (no power failure)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>ILP Eff. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mediabench</td>
<td>60%</td>
</tr>
<tr>
<td>Mibench</td>
<td>71%</td>
</tr>
<tr>
<td>Geomean</td>
<td>63%</td>
</tr>
</tbody>
</table>
Sensitivity to Cache Size

- NVCache
- NVSRAM
- WT-VCache
- ReplayCache

* Office power trace
Conclusion

• A pure software design for enabling WB volatile cache with crash consistency guarantee.

• Never amplify writes.

• Comparable performance to an ideal NVSRAM cache for realistic power traces.

Unpersisted stores in a power-interrupted region
Thank You

Q&A
Jianping Zeng @Purdue University
(zeng207@purdue.edu)

This presentation and recording belong to the authors. No distribution is allowed without the authors' permission.
Normalized Energy Consumption Breakdown

- NVCache
- NVSRAM
- WT-VCache
- ReplayCache

Breakdown [%]

- Cache
- Main Memory
- Compute

* Office power trace
Architecture of WT-VCache and WB NVCache

(a). Write-through (WT) volatile cache

(b). Write-back (WB) Nonvolatile cache
Speedup over Non-cache Baseline

- **NVCache**: 3.3 Home trace, 3.2 Office trace
- **NVSRAM**: 11.3 Home trace, 11 Office trace
- **WT-VCache**: 4.2 Home trace, 4.4 Office trace
- **ReplayCache**: 8.9 Home trace, 8.5 Office trace

54th IEEE/ACM International Symposium on Microarchitecture