CapOS: Capacitor Error Resilience for Energy Harvesting Systems

Jongouk Choi
University of Central Florida

Hyunwoo Joe
ETRI

Changhee Jung
Purdue University

I. Introduction

Energy-harvesting devices are a desirable replacement for battery-powered IoT. They utilize a capacitor as an energy buffer instead of a battery, gather ambient energy (such as RF radiation) there, then opportunistically use the buffered energy to power the devices. However, because ambient energy sources are unstable, energy harvesting systems frequently experience power outages during which all volatile data are destroyed, which causes program integrity to be compromised. In order to solve the issue, they use nonvolatile memory (NVM), which requires less power than a cache, as main memory. They also feature some sort of recovery support to back up and restore volatile data, such as registers, across a power outage.

Energy harvesting systems use a capacitor-backed just-in-time (JIT) checkpoint mechanism for recovery, similar to Intel's ADR memory subsystem [1]. They continue to monitor the level of buffered energy in the capacitor with a voltage monitor, and they checkpoint volatile data (registers) into NVM when a power loss is imminent. Then, in the wake of the outage, they restore the checkpointed registers from NVM and resume the program from the point of power interruption, with no rollback—a process known as roll-forward recovery. In this way, the JIT checkpointing ensures correct recovery and gives the systems an illusion that the volatile data are persistent across power failure.

Unfortunately, we discovered that capacitor degradation can incapacitate the capacitor-backed JIT checkpointing. Because of the frequent power outages, the capacitor is repeatedly charged and discharged, which is particularly stressful for the capacitor and accelerates its degradation. Furthermore, because energy harvesting systems are used as IoT devices in a variety of settings, their capacitors are frequently subjected to other stressful conditions, such as high humidity/temperature. Under these conditions, the capacitor degrades significantly over time and eventually becomes incapable of buffering enough energy for JIT checkpointing. When the degraded capacitor has insufficient buffered energy, energy harvesting systems fail the checkpointing, corrupting/losing volatile data (without providing any further service) across power failure; we refer to this as a *capacitor error*.

This paper introduces CapOS, a lightweight operating system (OS)-driven capacitor error resilience solution, to deal with the capacitor error. Depending on the condition of the capacitor, CapOS operates in either (1) normal mode or (2) safe mode. When the capacitor is not degraded, CapOS operates in normal mode, with JIT checkpointing acting as a roll-forward recovery mechanism. When a capacitor error is detected, CapOS enters a safe mode that performs a rollback recovery with JIT checkpointing disabled; while the degraded capacitor is idle, it restores the original capacitance on its own due to the resilient nature of the capacitor. When the capacitor is fully recovered, CapOS returns to normal mode. In sum, CapOS switches between the two modes when a capacitor error is detected.

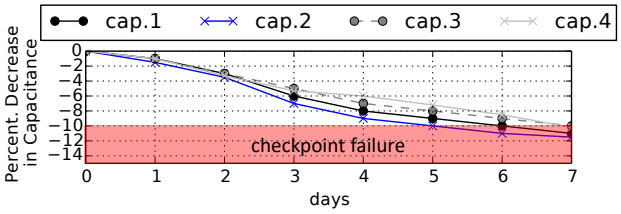


Fig. 1: Capacitor degradation in energy harvesting systems

II. Background and Motivation: Capacitor Error

Capacitor Error: To analyze the capacitor error in energy harvesting systems, we conducted experiments with a roll-forward recovery solution, Samoyed [2], having a 1mF supercapacitor as energy buffer on MSP430FR5994; we tested four different 1mF supercapacitors (e.g., cap.1∼4). For experiments, we developed the power generator board with MSP430FR5969 to incur power failures; the power generator supplies voltages to our target board through GPIO pins. Then, we mimicked energy harvesting situation by injecting square wave voltages into the target board.

From the experiments, we found that *the capacitors gradually lose its original capacitance* by more than 10% as shown in fig. 1 within seven days, which could cause a JIT checkpoint failure corrupting data in NVM, i.e., capacitor error. For further analysis, we also tested our real energy harvesting board (TIDA-00588), which is equipped with onboard solar cells with a 47mF capacitor as an energy buffer in a real harvesting environment. We found that the capacitance could be decreased by up to 50% within a year. Consequently, with the degraded capacitor, all the prior works suffer the capacitor error ensuring neither correct recovery (data loss) nor forward progress execution. Therefore, we believe there is a compelling need to address the capacitor error for practical energy harvesting systems.

Capacitor Recovery: Although capacitors are degraded in stressful conditions, a supercapacitor can particularly recover its capacitance when it is (electrically) isolated thanks to its self-recovery nature [3]. A prior work demonstrates the capacitor recovery phenomenon [3] with a recovery model.

To explore the capacitor self-recovery phenomenon in energy harvesting systems, we intentionally stressed supercapacitors to be degraded and then, when the capacitor was degraded by 10%, we electrically isolated them and measured their capacitance variation over time. Finally, we found that all of the degraded capacitors could be healed within 2 hours.

III. CapOS Design

To achieve a lightweight capacitor error resilience solution, CapOS dynamically detects the capacitor error in a reactive yet safe way. Since reactive error detection condones the JIT checkpointing failure corrupting the data, care must be taken to keep all the registers being checkpointed safe. For this purpose, CapOS leverages an acknowledgment (ACK) as a barrier of the JIT checkpointing, i.e., CapOS persists the ACK after checkpointing all registers in NVM. The insight is that a capacitor does not become faulty all of a sudden, but instead it is gradually degraded over time. That is, when the capacitor is first degraded, the JIT checkpointing only fails to write the last data, i.e., ACK, to NVM yet succeeds in persisting all registers. Consequently, if the capacitor error occurs, CapOS can detect it by checking the ACK in the wake of power failure.

Once the ACK corruption is detected, CapOS switches its execution mode from normal to safe. In the safe mode, CapOS disables the JIT checkpoint mechanism since the registers can eventually be corrupted as the faulty capacitor is further degraded. Then, CapOS electrically isolates the faulty capacitor to restore its original capacitance via the self-recovery nature of a supercapacitor [3]. To recover from power failure without JIT checkpointing, the safe mode needs another type of power failure recovery solution, i.e., rollback recovery.

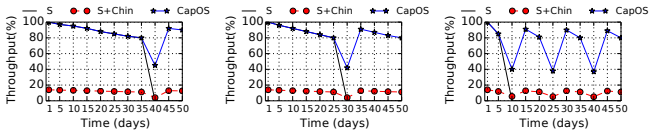
However, it is a daunting challenge to seamlessly switch from the roll-forward to the rollback and vice versa. This is because, unlike the roll-forward recovery, the program should be partitioned into recoverable regions, the boundary of which serves as a rollback recovery point in case the following region is interrupted by power failure. The problem is that although a capacitor error occurs at any given time, the traditional rollback recovery schemes cannot achieve seamless mode change because of their statically fixed region boundaries; they should enter the safe mode only at the beginning of a region—which would otherwise miss saving required data (memory logs and checkpoints) for the rollback recovery—though errors occur not necessarily at the region boundary.

To correctly enter the safe mode whenever the capacitor error is detected, CapOS leverages a boundary-free rollback recovery scheme comprised of timer-based checkpointing and copy-on-write (CoW) backed with a memory protection unit (MPU). CapOS can enter the safe mode at any point on which a recovery point is set with checkpointing all registers and starting a watchdog timer; upon the expiration of the timer, which serves as the new recovery point thus checkpointing the registers, a new checkpoint interval begins with restarting the timer. In case each interval is power-interrupted, the safe mode tracks memory updates during the interval using the MPU and logs the original pages with CoW giving them write permission—reclaimed at the beginning of an interval—to restore them with registers and restart the interrupted interval when power comes back.

When the capacitor is recovered, CapOS returns to the normal mode. To figure out when to return, CapOS leverages a capacitor recovery time model [3] and dummy JIT checkpointing. CapOS dynamically measures the capacitor isolation period and compares it with the model-estimated time to be taken for the recovery [3]. If the total isolation time exceeds the estimate, CapOS checks the capacitor by performing the JIT checkpointing of dummy data. Once the dummy JIT checkpointing succeeds, CapOS assures that the capacitor has become reliable and returns to the normal mode.

IV. Evaluation

We implemented CapOS, measured its throughput, and compared it to Samoyed (S) [2] and Samoyed with a reactive detection scheme, i.e., Chinchilla [4] (S+Chin.). We developed S+Chin as an alternative solution that can run rollback-enabled binaries [4] but enable the JIT checkpointing mechanism when the capacitor is stable; if the capacitor fails, the solution disables JIT checkpointing in the same way that CapOS does. We ran benchmark applications [4] on each scheme with collected three power traces from a real RF energy harvester for analysis. Figure 2 depicts the overall performance overhead of each solution. CapOS outperforms S+Chin by 4∼5x, while S fails to make progress within 30 days.



(a) Trace 1 (b) Trace 2 (c) Trace 3

Fig. 2: Performance analysis varying power failure patterns and recovery solutions

V. Summary

This paper found out that energy harvesting systems can fail checkpointing their volatile data to NVM due to the problem of capacitor degradation, which we call a capacitor error. To address the problem, we introduced CapOS, a lightweight and effective capacitor resilience solution. Our experiments demonstrate that CapOS can successfully address the capacitor error showing 4∼5x better throughput compared to a prior work on average.

References

[1] S. Scargall, *Programming Persistent Memory*. Intel, 2020.

[2] K. Maeng and B. Lucia, "Supporting peripherals in intermittent systems with just-in-time checkpoints," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 1101–1116, ACM, 2019.

[3] R. Chaari, O. Briat, and J.-M. Vinassa, "Capacitance recovery analysis and modelling of supercapacitors during cycling ageing tests," *Energy conversion and management*, vol. 82, pp. 37–45, 2014.

[4] K. Maeng and B. Lucia, "Adaptive dynamic checkpointing for safe efficient intermittent computing," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, (Carlsbad, CA), pp. 129–144, USENIX Association, 2018.