# FAQ/Appendix: A Survey on Checkpointing Techniques in Intermittent Systems

August 25, 2021

This FAQ answers some of the important doubts/questions that might arise while reading the survey. Some of the answers provide a detailed description of various points mentioned in the survey. Please note that this document contains additional information and is not a summary/gist of our survey. We suggest the reader go through the survey for an exhaustive and comprehensive study of the existing checkpointing approaches.

(**Q.1**): What novelty does the survey offer?

**Answer:** This survey performs a thorough comparison of various checkpointing approaches. Specifically, we implemented 13 state-of-the-art checkpointing approaches in an architectural simulator and rigorously compared them. We showed a detailed time and energy characteristics of different approaches. We discussed their relative strengths and weaknesses in great detail and made a set of concrete recommendations. To the best of our knowledge, such a rigorous experimental study has not been done before. We have also evaluated the sensitivity with respect to different capacitor sizes and ambient energy profiles. Finally, we have shown Kiviat plots that compare the approaches on a host of different metrics and can help the system designer decide which solution to choose.

In summary, the comparison provides us with dual benefits: (i) it tells the reader which classes of checkpointing approaches are the best, (ii) it shows the sensitivity of performance with respect to various external factors such as the nature of the energy source and the energy storage capacity.

**This is undeniably a novel and substantive contribution to the best of our knowledge.**

(**Q.2**): Considering that primary batteries can last for more than 30 years and over this period, they can generate much more energy than the EHD scenarios harvesting small amounts of power, then why are EHDs so favored?

**Answer:** We agree that a primary battery can have a reasonably high lifetime. However, various other factors should be considered while choosing the energy storage unit – batteries or capacitors.

(a) Large size of batteries: Typically, the energy density of lithium-based primary batteries is $0.8\mathrm{Wh\,cm^{-3}}$ [4]. Considering a lifetime of 30 years, this would result in a power density of 3µW cm$^{-3}$, which is much less than the power densities of various ambient sources (solar:15mW cm$^{-2}$, vibration: >50µW cm$^{-3}$, RF:50mW cm$^{-2}$). Thus, to power a device, the size of the battery would be much larger than the corresponding harvesting device. The large size of the batteries will make the device very bulky.

(b) Much less effective lifetime: There might exist batteries rated with a lifetime of around 30 years, but in practice, these batteries last for a much lesser time. The effective lifetime depends upon the duty cycle of the device using the battery and the environmental conditions under which the device runs. If the device is executing in continuous mode (i.e., 100% duty cycle), the lifetime would be much less than its prescribed lifetime. To run a device in a lesser than 100% duty cycle, we need to use a clock and a timer. This timer and clock require energy, and thus, this turns out to be a zero-sum game. Furthermore, a significant cause of reduction in the lifetime of a battery is *self-discharge* and *chemical decomposition*. An unused and packed battery can lose up to 8-20% of its original charge per year at a temperature of 20-30°$C$ [8]. These issues do not occur in energy harvesting-based devices.

In general, if we do not consider the device's size, the decision of choosing a battery or an ambient source depends upon the expected time for which the device will be used. As shown in Figure 0.1, the power density of a battery decreases linearly with its lifetime. In contrast, the power density is constant for an ambient source (for a particular ambient source and the harvesting conditions). A primary battery should be used if the device is to be run for a time period that is less than the intersection point (indicated by the dashed line in Figure 0.1); else, we should use an ambient source.

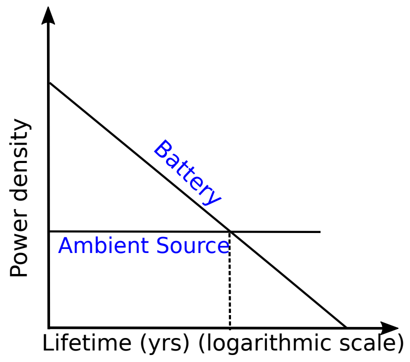

Figure 0.1: Power density vs. lifetime (adapted from [4]) of a battery and an ambient source.

Apart from these, the *cost* is yet another factor that favors energy harvesting over batteries. Though batteries are cheap, they are not preferred where a battery change is inconvenient due to the large volume of such replacements and the corresponding high costs. E.g., smart buildings that use EHDs can achieve 60% savings on the average annual costs [8]. In such cases, energy harvesting solutions are preferred.

(**Q.3**): One of the reasons behind the discouraged use of batteries is their harmful impact on the environment upon disposal. But can't batteries be recycled? Furthermore, the material used in energy harvesting devices is also often not environmentally friendly. These devices either use rare earth metals or have high levels of toxicity. Please comment on this discrepancy.

**Answer:** We agree that there are recycling programs to recycle substances such as lead, nickel, and mercury that batteries contain. However, as mentioned in [7], these substances are extremely expensive to recycle. Furthermore, recycling is not an entirely environment-friendly process; in fact, battery recycling programs are quite controversial. When the e-waste is sent to developing countries such as India, where labor is very cheap, recycling pollutes the environment and degrades the quality of human life.

EHDs are equal culprits in the sense that they also employ toxic rare earth metals; however, this is a broader debate, and such ethical issues are beyond the scope of this paper. Nevertheless, we would like to mention that EHDs fall in the same class as a large number of electronic components that use CMOS technologies. Their recycling industry is fairly mature, even though it is not perfect. At the end of the manufacturer, certain steps are also taken. For example, these poisonous elements are tightly encapsulated in compounds such as cadmium selenide and cadmium telluride, which themselves are environmentally stable. Hence, the toxic elements are highly unlikely to be released while the device is in use or upon disposal [5].

(**Q.4**): What data does a checkpoint (backup) include?

**Answer:** When the device's energy is about to exhaust, we backup the application state, which is composed of the memory state (includes the runtime stack, heap memory, and static data memory), register state (i.e., the CPU register file), peripheral state (both on and off-chip), and the program counter. The program counter could be a part of the CPU register file as in the case of the ARM architecture or can be a separate entity as in the x86 architecture. So, in general, we can consider the program counter a separate component of the application state.

(**Q.5**): What is state retention? Can SRAM be used for state retention?

**Answer:** State retention refers to storing the programâĂŹs state in some data storage element where it will remain intact with low or no power.

Yes, SRAM can be used for state retention. Jayakumar et al. [3] observed that with a voltage as low as $220mV$, SRAM cells could retain data for infinite time. Thus, in the low power mode, the SRAM can retain all its contents. Another observation made by Williams et al. [10] is that most intermittent systems have very short power-off times(<1s). After a power loss, the remaining charge in the device is sufficient to maintain the data in the SRAM array. Thus, in some cases where the power-off times are low, SRAM memory can be safely used for state retention.

(**Q.6**): Out-of-place state retention backs up the data, either off-chip or on-chip non volatile memory (NVM). Explain the approaches used for out-of-place state retention.

**Answer:** Off-chip checkpointing stores the backup in the external memory. Accessing the remote NVM is both energy and time-consuming. On-chip checkpointing gets the proximity benefits by using the NVM that has been integrated with the processor's CMOS circuits. However, using on-chip NVM is still not fully efficient as the data in the processor's flip-flops has to be copied to/from the centralized NVM. To solve this, nonvolatile processors (NVPs) have been proposed, which have nonvolatile flip-flops(NVFFs) [9]. Each NVFF is attached to a standard volatile flip-flop, enabling a parallel bit-to-bit transfer, resulting in energy and time efficiency [6]. However, this enhancement has an increased area cost due to the NVFFs and a slight increase in overall leakage energy.

(**Q.7**): In differential checkpointing, we checkpoint only the data that has changed after the previous checkpoint, rather than checkpointing the entire SRAM. Describe the various ways to compute this differential.

**Answer:** The various approaches to track the modified SRAM regions are as follows:

(a) Word-by-word comparison requires reading a large amount of NVM, i.e., equal to the SRAM size. Reading such a considerable amount of data is comparatively easy in Flash memory than in other byte-addressable NVMS such as FRAM. With Flash memory, reads are much cheaper than writes; thus, we can read and compare the blocks and then write only the modified data. In contrast, byte-addressable NVMs such as FRAM, where reads cost roughly the same as writes, could have directly written the entire SRAM. Thus, FRAM is not suitable for word-by-word comparison. Furthermore, performing block reads in Flash is cheaper than reading a block in byte-addressable NVMS (FRAM). We experimentally verified that performing a word-by-word comparison on $2KB$ of FRAM would take around 125µs, while it would roughly take roughly $50\mu s$ to read Flash memory. Thus, Flash memory is appropriate for this approach.

(b) Hash comparison: This approach does not require reading the entire NVM memory but only a few bytes, corresponding to the block's hash value. Therefore, this approach might be beneficial in the case of slow and power-hungry memory. However, due to the compute-intensive nature of the hash function, we also need to analyze its computation overhead. The summation of the compute and NVM overheads determines the total overheads of this approach. We ran the SHA-1 hash algorithm on our simulator to compute the energy and time overheads for computing the hash of 256 bytes of data. The experiment showed that computing one hash value consumed on average around 38.6μJ of energy and took around 1.7ms. Considering that FRAM writes take only 125ns for 2 bytes and FRAM writes are as fast as FRAM reads, the total time for reading the 20-byte hash digest for SHA-1 hash would only be 1.25μs. Hence, the hash computation overhead is prohibitive and is thus not the best choice for EHD-based systems.

(c) Tracking changes in volatile memory: Since this approach does not access the NVM to compute the changed data, this approach appears to be the least expensive. Furthermore, this approach has very low space and time overheads. It uses an in-memory bit array that results in a slight main memory overhead (at max 12.5%), and the bit array can be updated in constant time ($\mathcal{O}(1)$).

(**Q.8**)**:** The survey mentions three criteria - forward progress, correctness, and efficiency to compare various checkpointing approaches. Are all of these metrics important? Please comment on this.

**Answer:** Yes, all three criteria: forward progress, correctness, and efficiency are important. Among these, forward progress and correctness are necessary criteria that should be followed by all the checkpointing approaches, as they avoid non-termination and incorrect execution, respectively. Forward progress and correctness are non-negotiable; they are mandatory.

Efficiency, in contrast, is desirable. Ideally, an efficient checkpointing approach should (i) consume a very little amount of energy, (ii) take minimum time, (iii) have a small memory footprint, (iv) have minimal programmer intervention, and (v) have very infrequent re-executions; however, it is not often possible to satisfy all these requirements at the same time. Thus, we need to prioritize these requirements as per the application. Out of these, time and energy are the most important metrics that are considered in the EHD literature.

(**Q.9**)**:** In energy harvesting systems, memory consistency issues arise when an EHD writes directly to the NVM. If this possibility is removed, won't the memory consistency issues disappear?

**Answer:** Yes, if an EHD is not allowed to write directly to the NVM, the memory consistency issues can be avoided. However, preventing an EHD from writing data directly to the NVM is not preferred as it would make checkpointing more frequent (mandatory in some cases).

Checkpointing has its issues.

(a) The energy and time overheads of taking a checkpoint are proportional to the size of the program state [1]. Hence, frequent checkpointing is often not a scalable strategy.

(b) Taking a checkpoint is not always feasible, e.g., when the total energy of executing a code segment between two checkpoints and taking the checkpoint is more than the energy stored in the device's capacitor.

(c) Checkpointing requires halting the executing program.

(d) Sometimes we only need to track a few variables across failures, and checkpointing the entire system state is not preferred. Due to these reasons, applications need to have nonvolatile variables that are directly written to NVM memory.

(e) Please note that checkpointing could be avoided by storing all the required data in NVM. However, this is also not preferred since for each read/write operation, we need to access the slow and power-hungry NVM.

Hence, considering the two ends of the spectrum: (i) no nonvolatile variables and (ii) all variables stored in NVM, EHD applications prefer to have a system with mixed variables - volatile and nonvolatile. They basically find a hybrid strategy to be the most optimal.

(**Q.10**): Why does this survey compare the various checkpointing approaches in a simulator and not in actual hardware?

**Answer:** We have accurately simulated the microcontroller, considering the processor configuration and power values according to the device's datasheet. This is a standard approach (followed by existing works [2]) for doing research since these EHD devices are not always available easily, and their hardware implementation cannot be modified. Furthermore, when new hardware is proposed, it is practically infeasible to modify, fabricate, and characterize it. Therefore, simulation is a valid approach and is widely accepted in the whole embedded systems and computer architecture communities.

(**Q.11**): Please explain why does the survey paper follow the current organization? Also explain, Why have the existing checkpointing techniques been categorized in this way?

**Answer:**

- Organization: Our organization of various research proposals is based on their complexities, which roughly is in the chronological order in which the works have been proposed. E.g., In the past, let say 6-7 years ago, compiler-based approaches were more prevalent. Now, researchers are moving towards new areas such as learning-based approaches. We follow the same pattern and organize the survey based on the time (the year the main set of approaches were proposed) and complexity of the proposed approaches. Our approaches are built on top of each other, i.e., if two approaches are causally related, their order has been taken into account, and the parent approach is described first.
- Categorization: We looked at various existing survey papers and found how taxonomies are created. They all follow a standard practice: all the approaches are taken and clustered either by performance or design. Similarly, we initially clustered the approaches based on their design. When we analyzed and experimentally compared different approaches based

on performance, distinct clusters emerged, which were highly correlated to those created based on the design. Thus, our categorization is based on the standard practice of design philosophy and runtime characterization of the approaches. We basically formed "natural clusters" of approaches and discussed them together.

## References

[1] Alexei Colin and Brandon Lucia. Chain: Tasks and Channels for Reliable Intermittent Programs. In *Proceedings of the ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, 2016.

[2] Joakim Eriksson, Adam Dunkels, Niclas Finne, Fredrik Osterlind, and Thiemo Voigt. Mspsim–an extensible simulator for msp430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, volume 118, 2007.

[3] Hrishikesh Jayakumar, Arnab Raha, and Vijay Raghunathan. HYPNOS: An Ultra-Low Power Sleep Mode with SRAM Data Retention for Embedded Microcontrollers. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, pages 1–10, 2014.

[4] Maria Teresa Penella, Joan Albesa, and Manel Gasulla. Powering Wireless Sensor Nodes: Primary Batteries versus Energy Harvesting. In *2009 IEEE Instrumentation and Measurement Technology Conference*, pages 1625–1630. IEEE, 2009.

[5] Peter Harrop. Environmental issues with energy harvesting, 2009. `https://www.printedelectronicsworld.com/articles/1245/environmental-issues-with-energy-harvesting`.

[6] Jean-Michel Portal, Marc Bocquet, Mathieu Moreau, Hassen Aziza, Damien Deleruyelle, Yue Zhang, Wang Kang, Jacques-Olivier Klein, YG Zhang, Claude Chappert, et al. An Overview of Non-Volatile Flip-Flops Based on Emerging Memory Technologies. *Journal of Electronic Science and Technology*, 12(2):173–181, 2014.

[7] Runar Finanger. Here's why energy-harvesting trumps batteries. `https://www.onio.com/article/energy-harvesting-trumps-batteries.html`, 2020.

[8] Simon Aliwell. Batteries Not Enough - A Case for Energy Harvesting. `http://eh-network.org/resource6.php?id=128`, 2011.

[9] Fang Su, Kaisheng Ma, Xueqing Li, Tongda Wu, Yongpan Liu, and Vijaykrishnan Narayanan. Nonvolatile Processors: Why is it Trending? In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017.

[10] Harrison Williams, Xun Jian, and Matthew Hicks. Forget Failure: Exploiting SRAM Data Remanence for Low-overhead Intermittent Computation. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 69–84, 2020.