The Journal of Supercomputing (2022) 78:3425–3447
https://doi.org/10.1007/s11227-021-03999-8

Performance-aware cache management for energy-harvesting nonvolatile processors

Yan Wang[1] · Kenli Li[2] · Xia Deng[1] · Keqin Li[2,3]

Accepted: 12 July 2021 / Published online: 26 July 2021

Abstract
With the increasing popularity of wearable, implantable, and Internet of Things devices, energy-harvesting nonvolatile processors (NVPs) have become promising alternative platforms due to their durability when running on an intermittent power supply. To address the problem of an intermittent power supply, backing up of volatile data into a nonvolatile cache has been proposed to avoid the frequent need to restart the program from the beginning. However, the penalties incurred by frequent backup and recovery operations significantly degrade the system performance and waste considerable energy resources. Moreover, the increasing amounts of data to be processed pose critical challenges in energy-harvesting NVP platforms with tight energy and latency budgets. To further improve the performance of NVPs, this article adopts a retention state that can enable a system to retain data in a volatile cache to wait for power recovery instead of backing up data immediately. Based on the retention time, we propose a performance-aware cache management scheme and a pre-backup method to improve the system performance and energy utilization while guaranteeing successful backup. The pre-backup method is also optimized by retaining data in a volatile cache when receiving a high voltage warning. In particular, the nonvolatile memory (NVM) compression technique is introduced to achieve the goal of minimizing power failures and maximizing system performance. Moreover, the security problems in the sleep state are discussed with regard to the NVM compression technique to guarantee the NVP's security. We evaluate the performance and energy consumption of our proposed algorithms in comparison with the dual-threshold scheme. The experimental results show that compared with the dual-threshold scheme, the proposed algorithms together can achieve a 52.6% energy reduction and a 13.72% performance improvement on average.

Keywords Backup · Energy harvesting · High performance · Nonvolatile processors (NVPs)

✉ Yan Wang
yanw@gzhu.edu.cn

Extended author information available on the last page of the article

1 Introduction

With the advent of the internet of things (IoT), implantable and wearable medical devices have been developed to maintain close contact with users to facilitate responsiveness to their needs. These small wearable devices are no longer suitable for running on battery power [26] because of various disadvantages presented by batteries, such as their large size and weight and their need for frequent charging. As alternatives, energy-harvesting techniques are promising means to power IoT devices in place of batteries, since energy-harvesting devices can harvest energy to continue working properly without frequent maintenance [18]. Energy-harvesting systems harness energy from ambient energy sources such as wind, solar, and electromagnetic radiation and from thermal and vibration energy [8].

However, energy harvesting is susceptible to the problem of power source instability. The system will often be interrupted and rolled back in the event of power failure and recovery. Frequent and unpredictable power interruptions cause traditional processors to suffer from either many operating rollbacks or large backup overheads, hindering the completion of large tasks and incurring a substantial energy cost. Fortunately, nonvolatile processors (NVPs) have been proposed to solve this problem. When a power outage occurs, NVPs can use the energy stored in the capacitor to back up volatile data to a nonvolatile cache, then take a snapshot of the programs being executed. When the power resumes, the processor recalls the pre-stored data so that the program execution can be continued [20]. To guarantee the performance of the system, all applications that are executed on an NVP must be correctly recovered as fast as possible. Together with the increased demand for low-energy consumption, the high-performance cache management and backup problems in energy-harvesting NVP platforms have also become increasingly critical. Backup techniques on energy-harvesting NVP platforms have considerably improved system performance and reduced energy consumption, provide more opportunities for finer tuning of shorter execution times and energy utilization for programs, and pose new challenges to the research community [4]. Therefore, in this article, we consider an effective backup strategy for the target energy-harvesting NVP platform.

When NVPs are not stable, a poor backup strategy may lead to data consistency problems, and frequent data backup and recovery waste considerable energy and system resources and slow program execution [19]. Moreover, when an NVP takes a snapshot, a smart attacker could trigger instability, and it is possible that other, unknown security issues may yet emerge [6]. In addition, the backup caches for NVPs use nonvolatile memory (NVM), which suffers from high energy consumption and a high write latency; these drawbacks have impeded their widespread adoption. Therefore, when a backup strategy is executed on an energy-harvesting NVP platform, the following requirements must be satisfied. First, to guarantee the performance of the system, it must be possible to reduce unnecessary backup and recovery operations while guaranteeing that programs can be rapidly and correctly recovered. Second, to improve energy efficiency and utilization, the switching overhead, recovery overhead, and access overhead should be reduced as much as possible. Third, to minimize the execution time, memory latency should be avoided as much as possible, which implies that cache hit/miss and reads/writes should be carefully considered in cache management. Fourth, to enhance security, errors that occur during data saving or transmission due to external factors should be reduced, and the challenges of the nonvolatility, the lack of time keeping, and potential power attacks should be addressed.

Considering the requirements of a backup strategy for energy-harvesting NVPs, this paper will explore a performance-aware cache management and pre-backup strategy with the objective of minimizing unnecessary frequent data backups and recoveries, which directly waste considerable energy and system resources, slowing the program execution. In addition, considering the nature of NVPs, traditional encryption algorithms are not suitable for NVP systems because these algorithms can lead to high encryption and decryption overheads, which can reduce system performance [9]. Therefore, this paper will adopt an NVM compression technique for the nonvolatile part of the cache based on selective encryption [9] to improve NVP security and reduce the number of unnecessary read and write operations. The contributions of this work are summarized as follows.

- To minimize data access and migration, we introduce a data state mechanism based on retention time and present a performance-aware cache management, which considers the cache reads/writes, hit/miss, and data state.
- To address the drawback of the waste caused by frequent data backups and recoveries, combined with various voltage levels, we propose a two-step backup strategy: a pre-backup strategy triggered by a high voltage warning and a backup strategy triggered by a low voltage warning.
- The encryption/decryption methods will lead to performance degradation. To address the security-related problems, the corresponding solution of an NVM compression algorithm based on selective encryption is presented.

The remainder of this paper is organized as follows. Section 2 reviews the background of the study and summarizes the motivation for our proposed algorithms. Section 3 presents our target architecture model. Section 4 introduces the proposed performance-aware cache management scheme and an NVM compression technique in detail. In Sect. 5, we evaluate and analyze the proposed algorithms. Section 6 concludes the paper and discusses future work.

2 Related work and motivation

2.1 Backup for NVPs

Because NVPs behave differently from traditional processors, the backup strategy needs to cover the entire cache hierarchy and should be combined with effective policies for adaptive cache management. The main categories of backup strategies are (1) backup logic design [7, 13], (2) optimization of the backup procedure [16, 23], (3) backup-aware system management [17, 20], and (4) residual energy detection [6, 11]. Comprehensive studies have been conducted on NVP backup strategies. By incorporating access latency and recovery overhead into an energy-harvesting NVP, the energy consumption-driven backup problem has been studied in [21] and [22]. Zhou et al. [25] explored backup strategies and cache management policies in relation to the cache architecture for NVPs. The authors analyzed hybrid cache structures and power traces, then proposed energy-aware morphable cache management policies that could improve system performance and energy utilization while guaranteeing successful backup. Li et al. [12] proposed stack allocation and management policies to minimize backup in the stack while guaranteeing successful backup with limited available energy.

Although the above backup techniques have considerably improved system performance and reduced energy consumption, they are direct backup strategies. When a power shutoff occurs, the conventional direct backup strategies backup data in nonvolatile memory immediately. This will lead to a serious drawback in which the frequent unnecessary data backups and recoveries waste considerable energy and system resources, resulting in performance degradation. To overcome the serious drawback of frequent data backup and recovery operations, Zhou et al. [24] proposed a dual-threshold scheme that allows the system to be put into a sleep state to wait for power resumption when the voltage falls below a certain retention threshold instead of backing up data directly upon a power interruption.

Table 1 shows a high-level comparison among existing backup schemes and the pre-backup scheme in terms of frequent backup, unnecessary backup and energy utilization. In summary, the direct backup strategies incur frequent and unnecessary backups, which are not supportive of the NVP performance and energy utilization. While it does not cause frequent and unnecessary backups and has low energy utilization, the dual-threshold scheme has low cache utilization. The cache utilization is a very important parameter, especially in the context of space-limited NVM caches. Hence, it is important to design a performance-aware cache management scheme to minimize the NVM writes while ensuring high performance in terms of cache utilization and request latency. In this paper, based on the performance-aware cache management, we discuss a two-step backup strategy: a pre-backup algorithm and the backup strategy. We set two voltage warning thresholds. In response to a high voltage warning, the pre-backup algorithm is performed, and the backup strategy is performed in response to a low voltage warning.

2.2 Impact of retention time

In further optimization, we introduce retention state [10]. In accordance with the concept of a retention state, a retention threshold and a backup threshold can be established to allow a system to maintain volatile blocks for a certain period of time

Table 1 Comparisons of different backup strategies

Backup strategy	Frequent backup	Energy utilization	Cache utilization
Direct backup	High	Low	Low
Dual-threshold	Low	High	Low
Pre-backup	Low	High	High

to wait for power to be restored instead of performing immediate backup. When the voltage drops, capacitors are used as an energy source in the system [15]. If the voltage drops to the retention (high) threshold, the system will prepare for backup but maintain volatile blocks to wait for power restoration. If the system recovers in the retention state, no backup operation is necessary for the system to continue to run. However, if the voltage drops to the low threshold, the system will perform backup operations. In this way, a large number of backup operations can be avoided if the power supply to the energy-harvesting processors is quickly restored. In [10, 14], the authors analyzed the inter-write times (refresh times) of the level 1 (L1) and level 2 (L2) cache blocks by conducting an application-driven study and determined a suitable data retention time accordingly. Their application-driven analysis showed that the ideal retention time should be in the range of *ms*. This paper introduces a data state mechanism to facilitate effective cache management and data migration based on the retention time concept for NVPs.

2.3 NVM and NVP security

IoT technology trends and wearable devices innovations continue to exacerbate the performance gap between processors and memory. Cache memories have long been used to reduce average memory latency and bandwidth. Current energy-harvesting NVP systems provide two levels of static random-access memory (SRAM) caches and backup NVM caches. The field of NVM security is closely linked to that of NVM endurance. If a single NVM cell becomes unreliable in the case of a nonvolatile cache, the entire nonvolatile cache becomes useless. Another limitation is that programming an NVM cell consumes a large amount of power. Thus, the limited cache resources can be effectively organized to improve cache performance and NVP security for many memory-intensive commercial workloads. Cache compression is one way to improve the effectiveness of cache memories. Therefore, this paper proposes an NVM compression algorithm for the NVM part of the L2 cache to improve system performance.

The adoption of NVPs presents new security issues. Because an NVP can sleep indefinitely while the power is off and then continue its work as soon as the voltage is restored, a lack of power is harmless, and the entire memory content is persistent; however, this opens a door for an attacker to access the stored data [6]. Moreover, traditional encryption algorithms are not suitable for NVPs. This is because encryption in the cache would cause a large number of bit flips and cost considerable time and energy, resulting in degraded system performance and lifetime. One way to improve security and ensure performance is to use compression, as this reduces the size of each data block [9]. Consequently, attackers cannot obtain the original data, and the nonvolatile cache can simultaneously back up more data, thereby improving system performance. Therefore, an NVM compression algorithm for the NVM part of the L2 cache is proposed to improve system performance while guaranteeing system security.

2.4 Motivation

Motivated by the retention state concept and the need to resolve the security problems posed by NVPs, this paper considers an NVM compression technique combined with a four-stage performance-aware cache management scheme based on the traditional structure of NVPs. In this paper, we will clarify the problem by answering the following questions: (1) why do NVPs need to be backed up? (2) why do NVPs need to be secured? (3) why do backup operations and efforts to secure NVPs degrade system performance?

Thus, this paper addresses the following considerations for performance-aware cache management:

- Performance: because the access cost of static random-access memory (SRAM) is lower than that of NVM, SRAM is better suited for data caching and for writing sensitive data, whereas NVM is more suitable for reading sensitive data.
- Backup: because dirty volatile blocks need to be backed up to the NVM cache when a power outage occurs, to reduce the backup overhead, a large amount of data is expected to reside in the NVM part of the cache. It is necessary to ensure that there is sufficient space in the NVM part of the L2 cache for the data from the L1 cache and the SRAM part of the L2 cache to be successfully backed up.
- Security: backing up data when a power shutoff occurs can give rise to security threats. Thus, changing the original form of the data by, for example, compressing the backed-up data can improve security.
- Memory efficiency: the access latency between the memory and the CPU is very high. Thus, when a power shutoff occurs, a large amount of data is expected to remain in the NVM cache. Compression techniques can be used to increase effective L2 cache capacity in order to reduce L2 misses.

All of the above considerations directly impact the performance, security, and energy consumption of an NVP system; thus, these considerations need to be simultaneously addressed to achieve the goals of performance-aware cache management. In this article, we propose a four-stage performance-aware cache management scheme to address the problem at hand.

3 Architecture and model

3.1 Energy-harvesting NVP

Figure 1 illustrates the target architecture model, which shows the relationship among the three different levels of the memory hierarchy: (a) the L1 cache; (b) the hybrid L2 cache, including an NVM part and a SRAM part; and (c) the nonvolatile main memory (NVMM). In this architecture, data can be transferred between adjacent layers of memory. For example, data can be transferred from the L1 cache to the NVM part or the SRAM part of the L2 cache but not directly to the NVMM.

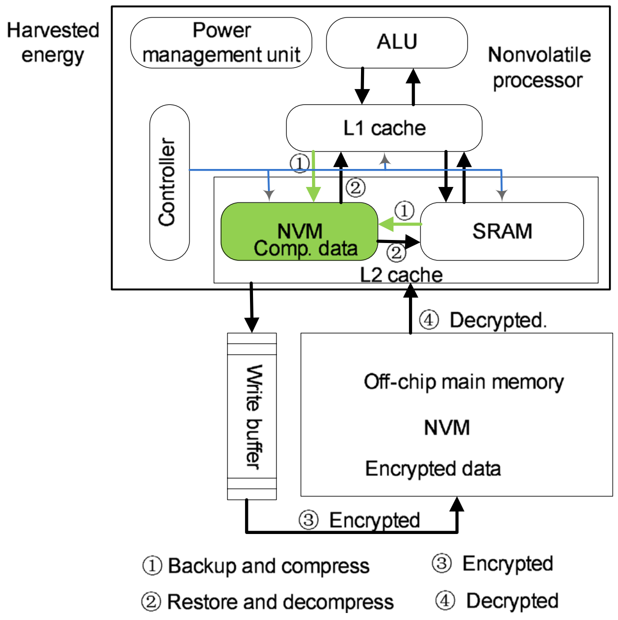


Fig. 1 Architecture model of the memory hierarchy

The NVM part of the hybrid L2 cache is a cached backup component; when a power shutoff occurs, the data located in the L1 cache and the SRAM part of the L2 cache can be backed up to the NVM part of the L2 cache. When power returns, the data are restored from the NVM part of the L2 cache to the SRAM part of the L2 cache and the L1 cache. As shown in Fig. 1, the controller in the NVP controls and manages the data transfer between the L1 and L2 caches. This controller is connected to the power management unit (PMU) and can be used to control the backup and recovery of the data depending on the voltage.

3.2 Data state of a volatile cache

To manage and identify the cache blocks effectively, we establish a counter for each cache block to track its state. In the data state mechanism, each cache block has an n-bit counter; thus, each cache block has 2^n possible states, as shown in Fig. 2. After a time T, the state of the cache block transitions from one state to the next. We assume that the transition time T is equal to the retention time divided by the number of states, i.e., $T = \frac{\text{(retention time)}}{2^n}$, where *retention time* is the time interval between two successive refreshes to the same cache block. Increasing the size of the counter will decrease the transition time at the expense of checking the blocks at a finer granularity cost. A cache block starts in state S_0 when it is accessed to the L1 cache. After every time interval T, the state of each valid cache block is incremented. However, any time a cache block is updated, the state of the cache block transitions from the current state back to the starting state S_0. When a cache block reaches the state S_{n-1},

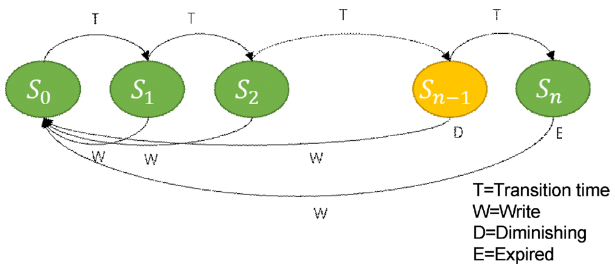


Fig. 2 The states of a cache block

then it will expire after another transition time T elapses. We define a cache block in state S_{n-1} as a *diminishing block* and a cache block in state S_n as an *expired block*. An *expired block* will be written to the L2 cache and marked invalid, and it can then be replaced by a new cache block in state S_0. For example, for a 2-bit counter, a valid cache block can be in one of four states, from state S_0 to S_3. The cache block transitions from state S_k to state S_{k+1} after 2.5 ms, and when the block is updated, it returns to the initial state S_0.

This cache block state mechanism is also suitable for the SRAM part of the L2 cache. When a cache block is updated, it returns to the initial state S_0. When a block reaches state S_{n-1}, it will be backed up into the NVM part of the L2 cache and marked invalid.

4 Methodology

The performance-aware cache management scheme consists of four stages with different actions, which are taken depending on the remaining energy of the power pulses. When power is restored, the goal is to resume execution (Stage 1). In Stage 2, the main goal of cache management is to achieve better performance because sufficient energy is available; thus, performance-aware data migration is performed. If the voltage drops to a predefined retention threshold, space is cleared in the NVM part of the L2 cache to prepare for backup, and the cache is then put into a sleep state (Stage 3). Stage 4 corresponds to the case in which the voltage drops further down to the backup threshold, meaning that backup operations should be performed.

4.1 Performance-aware data migration

In Stage 2, the NVP has sufficient energy, so efficient cache management is prioritized, taking advantage of both the NVM and SRAM parts of the L2 cache as well as the L1 cache. Cache management in this stage mainly focuses on the following three considerations. First, the impact of the retention time, i.e., when a cache block reaches state S_{n-1}, on the system performance should be optimized to present the data that have expired. Second, blocks with write-sensitive data should reside in the SRAM part of the L2 cache. Third, blocks with read-sensitive data should be located in the NVM part of the L2 cache. To identify the cache behaviors, a counter is set for each cache block to track the block's state. These considerations are used to guide the performance-aware data migration algorithm (Algorithm 1).

Algorithm 1 Performance-aware data migration

```
Input: (1) cache blocks, (2) n-bit counters
 1: for each cache block in the L1 cache in state $S_{n-1}$ do
 2:    if the cache block is dirty then
 3:       write it into the SRAM part of the L2 cache;
 4:       mark it invalid;
 5:    else
 6:       if $index == 1$ then
 7:          reset $state = S_0$ and $index = 0$;
 8:       else
 9:          mark it invalid;
10:       end if
11:    end if
12: end for
13: for each cache block in the SRAM part of L2 cache do
14:    if perform a read operation for the block then
15:       $nr++$
16:    end if
17:    if read operation $nr$ is larger than $rt$ then
18:       migrate it to the NVM part of the L2 cache;
19:       nr=0
20:    end if
21:    if state is $S_{n-1}$ then
22:       reset state as $S_0$
23:    end if
24: end for
25: for each cache block in the NVM part of L2 cache do
26:    if perform an update or a write operation then
27:       $wr++$
28:    end if
29:    if write operations $wr$ is greater than $wt$ then
30:       find an SRAM block with the largest read operations
          $nr$ and swap its contents with those of this block
31:    end if
32: end for
```

As shown in Algorithm 1, if the state bit of an L1 cache block becomes S_n, which means that it will disappear from the L1 cache, the *dirty bit* and the *index bit* are checked, where a dirty bit denotes whether the cache block has been updated and the index denotes whether the block has been used recently. If the block is dirty, it needs to be written to the SRAM and marked as invalid (lines 2–4). If it is not a dirty block but the *index bit* is 1, which means that the block is frequently used, the block will be returned to its starting state (lines 6–7). Otherwise, this block is directly marked as invalid (line 9). For the L2 cache, to ensure write-sensitive data can reside in the SRAM part and read-sensitive data can reside in the NVM part, we set a write threshold *wt* for the NVM part and a read threshold *wt* for the SRAM part. The corresponding variables *wr* and *nr* are set to record the write numbers and read number for each NVM block and SRAM block, respectively (lines 14–16, 26–28). For a block in the SRAM part of the L2 cache, if the read number *nr* is greater than the read threshold *rt*, it needs to be migrated to the NVM part (lines 17-19). To reduce the elimination of SRAM blocks, if the state is S_{n-1}, we will reset it as S_0 (lines 21–23). For a block in the NVM part of the L2 cache, if the write record *wr* is larger than write threshold *wt*, it needs to be migrated to the volatile part of the L2 cache. If the SRAM part has invalid blocks, the block is migrated into the SRAM part. Otherwise, the block with largest read number *nr* is found, and its contents are swapped with the contents of this block (lines 29–31).

4.2 Pre-backup techniques

In this article, to minimize unnecessary backups, we adopt an improved version of the dual-threshold scheme [24], and based on the improved version, we propose a pre-backup method. Specifically, we set two voltage thresholds, namely, a low voltage threshold V_{Low} and a high voltage threshold V_{High}. When the voltage falls below the high voltage threshold V_{High}, we will perform pre-backup operations, and when the voltage drops below the low voltage threshold, the data in the L1 cache and the SRAM part of the L2 cache will be backed up to the NVM part of the L2 cache. The threshold V_{High} is set as follows.

High voltage threshold V_{High}: This threshold is a retention threshold at which the system enters the sleep state to wait for power recovery instead of backing up immediately. On the basis of this retention threshold, volatile cache data will remain in the cache for a certain period of time during an outage. Thus, the high voltage threshold V_{High} determines how long the system can remain in the sleep state before backing up. However, there is no guaranteed sufficient time to wait for energy resumption. Therefore, to ensure that the data can be successfully backed up, when the voltage drops below the high voltage threshold, the system should write some NVM blocks to prepare for backup. Therefore, the required energy consumption is as follows:

$$E_{High} = N_{NVM} \times E_{write} + N_{NVM} \times E_{comp} + E_{sleep} + E_{low}, \tag{1}$$

where N_{NVM} denotes the number of NVM blocks in the L2 cache, E_{write} denotes the energy consumption for writing an NVM block to the main memory, E_{comp} denotes the energy consumption for the compression of an NVM block, E_{sleep} is the energy consumed in the hibernation state, and E_{low} is the required energy for the low voltage threshold. Thus, the high voltage threshold V_{High} is as follows:

$$V_{High} = \sqrt{\frac{2 \times E_{High}}{C}}. \tag{2}$$

The amount of energy stored in a capacitor has nothing to do with the length of time but rather with its own capacity and the actual voltage between its two pins. The formula is $E = \frac{1}{2} * C * V^2$ [24].

The pre-backup operations are described in Algorithm 2. When the energy drops to the high voltage threshold V_{High}, the system should prepare for a power failure, and the pre-backup operation should be performed. At this time, the processor does not immediately stop running programs; instead, nonvolatile blocks are cleared, and the system goes into sleep mode, ensuring that there is sufficient space for backup and properly backing up some appropriate data. For safety, it is assumed that in the worst case, all SRAM blocks are dirty and that the NVM part of the L2 cache can hold all of them. Thus, the number of NVM blocks in the L2 cache that need to be reserved for backup is equal to the number of SRAM blocks (line 2). We first calculate the free size of the NVM after compression; if the free space is not sufficient for backup, the system will sort the NVM blocks by LRU status. Then, the most recently unused NVM blocks should be cleared by writing them to the main memory (lines 3–13). To ensure the volatile blocks cannot be eliminated during the retention state, the volatile blocks of the L2 cache with state S_{n-1} will be migrated into the NVM part (lines 14–19). In addition, the dirty L1 cache block with state S_{n-1} will be written into the NVM part (lines 20–25).

Algorithm 2 Pre-backup operations

```
Input: (1) the dirty bit of each cache block in the L2 cache,
    (2) each cache block in the SRAM in state $S_{n-1}$
 1: if the voltage $V \leq V_{high}$ then
 2:    $m$ =SRAM block/2;
 3:    determine the free space of NVM part→ $Space$;
 4:    if $Space < m$ then
 5:       for each valid NVM block in the LRU list in order
          of decreasing age do
 6:          write it back to the main memory;
 7:          $Space = Space + 1$;
 8:          mark it invalid;
 9:          if $Space == m$ then
10:             break;
11:          end if
12:       end for
13:    end if
14:    for all SRAM blocks of L2 cache do
15:       if the state is $S_{n-1}$ then
16:          migrate the block to the NVM part of the L2
             cache;
17:          mark it invalid;
18:       end if
19:    end for
20:    for all L1 cache blocks do
21:       if the block is dirty and the state is $S_{n-1}$ then
22:          write the block back to the NVM part of the L2
             cache;
23:          mark it invalid;
24:       end if
25:    end for
26: end if
```

4.3 Backup

When a power warning is received, the remaining available energy in the capacitor is compared with the required low energy consumption. If the available energy exceeds the energy required, the program can perform pre-backup operations and continue until the next energy warning is received. If the available voltage drops to the required low voltage threshold V_{Low}, the backup warning is triggered. The dirty blocks in the L1 cache and the SRAM part of the L2 cache are backed up to the NVM part of the L2 cache. The low voltage threshold V_{Low} is set as follows:

Low voltage threshold V_{Low}: to ensure adequate energy for backup, the worst-case scenario is considered, in which all SRAM blocks need to be backed up; thus, the required energy consumption is as follows:

$$E_{low} = N_{SRAM} \times E_{write\_to\_NVM}, \tag{3}$$

where N_{SRAM} denotes the number of SRAM blocks in the L2 cache and $E_{write\_to\_NVM}$ denotes the energy consumption for backing up an SRAM block. Thus, the low voltage threshold V_{Low} is as follows:

$$V_{Low} = \sqrt{\frac{2 \times E_{low}}{C}}, \tag{4}$$

where C is the capacitance of the capacitor.

The performance-aware cache management scheme dynamically performs a backup on the basis of the data state, voltage, and cache behaviors to utilize retention time appropriately in a manner that improves performance while guaranteeing successful backups. The backup operations are as follows: when the low voltage warning is received, we first backup all valid blocks in the SRAM part to the NVM part; then, the dirty L1 cache blocks are written to the NVM part.

4.4 NVM cache compression

For enhanced backup requirement and performance, we propose a compression technique used in the NVM part of the L2 cache. Compression increases the effective NVM capacity of the L2 cache in order to reduce L2 misses, effectively increasing the performance of the NVM part of the L2 cache. However, while compression helps protect the NVP and eliminate long-latency L2 misses, it also increases the latency overhead of decompression. Thus, we propose an adaptive compression technique, shown in Algorithm 3, to compress cache blocks effectively on the basis of the voltage and the historical use of the cache. In the algorithm, we use a constant *segment_com* to indicate the number of data blocks that can be held in a compression NVM block, and we use a binary variable *comflag* to represent the compression state in an NVM block, where *comflag* $= 1$ means that the NVM block is a compression block and can hold *segment_com* $= N$ data block.

Algorithm 3 NVM compression

```
Input: cache blocks in the NVM
1: /*all blocks in the NVM are compressed when the power
   is on*/
2: while the energy is sufficient do
3:    for each block do
4:       while load a data from main memory do
5:          if the block is compressed then
6:             if $\frac{N_{hit}}{N_{miss}} \geq \frac{T_{miss}}{T_{decom}}$ then
7:                set compression state $comflag = 0$
8:                set $segment\_com = 1$
9:                access the new data to the block
10:            else
11:               access and compress the new data into the
                  block
12:            end if
13:         else
14:            if $\frac{N_{miss}}{N_{hit}} \geq \frac{T_{decom}}{T_{miss}}$ then
15:               set compression state $comflag = 1$
16:               set $segment\_com = N$
17:               access and compress the new data into the
                  block
18:            else
19:               access the new data to the block
20:            end if
21:         end if
22:      end while
23:   end for
24: end while
25: if the voltage is less than the retention threshold then
26:    compress the cache block;
27: end if
28: determine the free space in the NVM;
```

To enhance security, when the power is off, the data in the NVM part of the L2 cache need to be compressed. Regarding performance, when the power is sufficient, some blocks (or benchmarks) will benefit from compression, but others will suffer. Thus, when loading data from main memory to NVM, we have to determine whether the data need to be compressed. For simplicity, we count the numbers of hits and misses and consider the decompression time for each NVM block, yielding (lines 5–10) the following:

$$\frac{N_{hit}}{N_{miss}} \geq \frac{T_{miss}}{T_{decom}},$$

where N_{miss} is the number of misses in an NVM block, N_{hit} is the number of hits in an NVM block, T_{decom} is the time required for the decompression of an NVM block, and T_{miss} is the time required to access data from the main memory. If the above equation is satisfied, the compression block should be decompressed. Then, we set *comflag* = 0 and access the new data block into the NVM block (the NVM block only holds the new data block). For a 6-cycle decompression penalty and 120-cycle NVM miss penalty, compression wins if it eliminates at least one NVM miss for every $120/6 = 20$ penalized NVM hits. Otherwise, we access and compress the new data into the NVM block.

For decompression blocks, block compression will be advantageous if (lines 14–18)

$$\frac{N_{miss}}{N_{hit}} \geq \frac{T_{decom}}{T_{miss}}.$$

If compression is advantageous, both the old data block in the NVM block and new data accessed from main memory are compressed in the NVM block. The compression state *comflag* will be set as 1. If the voltage is less than the retention threshold, all blocks in the NVM are compressed, as are the blocks that are newly migrated into the NVM.

To improve the security level of the nonvolatile main memory, the idea is to use an encryption algorithm to encrypt the compressed data. In addition, the system performance and lifetime of the nonvolatile main memory can be improved by the use of selective encryption [9], which can be introduced to solve the security problem of nonvolatile main memory.

5 Experiment

In this section, we will present the experimental setup and evaluate the proposed algorithms.

5.1 Experimental setup

In this experiment, the target architecture is an energy-harvesting NVP. The detailed system configuration is shown in Table 2. In each hybrid L2 cache set, there are three FREM blocks and four SRAM blocks. The detailed cache characteristics are shown in Table 3. In the experiment, these system parameters are fed into the gem5 simulator [5] to build an energy-harvesting NVP platform. The benchmarks are obtained from MiBench [2] and Embench [1], whose program sizes are suitable for representative application-specific IoT/embedded systems. Table 4 lists the detailed information of each benchmark. The energy harvesting and consumption can be accurately simulated and monitored. A portion of the power pulses that we used were extracted from a real solar database, namely, the Measurement and Instrumentation Data Center [3]. To devise a working environment for our simulator that was comprehensively similar to the real world, we used two energy traces, as shown in Fig. 3. The first trace is radio frequency (RF) energy, and the second is solar energy. To ensure the capacitor can hold enough energy to back up all of the volatile data upon receipt of a low voltage warning, we set the capacitor size to 50 ÌF.

We compare the proposed algorithm with the dual-threshold scheme (abbreviated as dual-th) [24], which is a recently published algorithm for minimizing backup operations by considering a retention state. We evaluate the energy consumption and performance (run time) of both strategies. The energy consumption comprises both static and dynamic energy consumption. In this paper, the dual-threshold scheme has been developed such that it is comparable to our model. We implement both strategies in the same scheduling framework to ensure that differences in implementation do not incur performance losses of the dual-threshold scheme.

5.2 Results and analysis

This section presents the experimental results to illustrate the effectiveness of our proposed algorithms. In the experiment, the dual-threshold scheme was taken as the baseline algorithm; the performance-aware cache management scheme, which included performance-aware data allocation (Algorithm 1) and pre-backup operations (Algorithm 2), was applied alone (denoted by *PACM*), and the full set of proposed algorithms was applied by combining the performance-aware cache management scheme with the NVM compression algorithm. The three strategies adopted the same backup method. When a backup warning is received, the L1 cache blocks with a dirty state are written to FREM, and all valid SRAM blocks are backed up to FREM.

5.2.1 Performance results

Table 5 shows the performance results for the three strategies under the RF energy traces. From the table, we can see that the PACM scheme exhibits slightly better performance than the dual-threshold scheme, while the proposed algorithms together exhibit an obvious performance enhancement. For example, for the JPEG benchmark performance of the NVP with a high voltage threshold 1.28 V, the execution time of the dual-threshold algorithm, 3.12×10^{11}, is larger than that of the PAMC scheme, at 3.08×10^{11}, and the execution time of our proposed algorithm is 2.98×10^{11}, which is superior to those of the dual-threshold and PAMC

Table 2 System configuration

Component	Description
Processor	1.0 GHZ Fetch/Exec/Commit, 1 core
L1 cache (SRAM)	4 kB per core (private) I/D cache, 64 B block size, write-back, 10 MSHRs, 2-cycle latency
L2 cache (hybrid SRAM and FREM)	16 kB SRAM, 16 kB FREM, 64 B block size, LRU; write-back
Main memory	120-cycle access time

Table 3 Cache characteristics

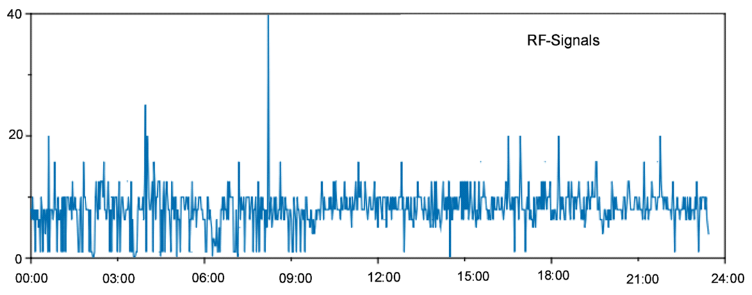
Cost	L1 cache	L2 cache	
		SRAM	FREM
Read latency (cycle)	2.89	7.12	9.08
Write latency (cycle)	2.34	6.89	20.12
Read energy (nJ)	0.118	0.161	0.216
Write energy (nJ)	0.112	0.156	0.839

Table 4 Benchmarks

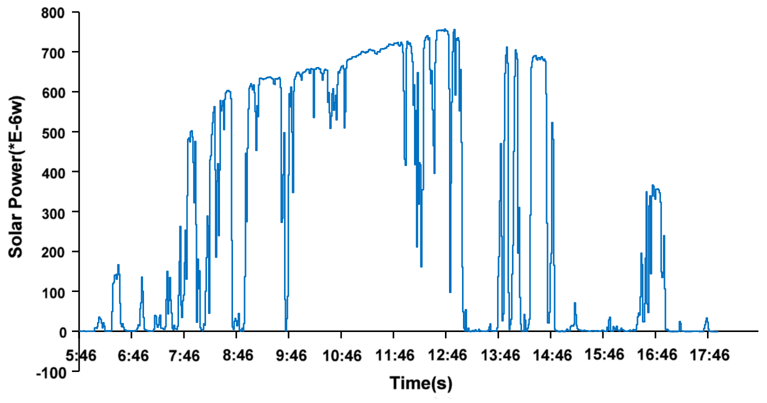
Benchmark	Instructions	Mem reads	Mem writes	Code size (Kb)	Data size (Kb)
jpeg	43,487,772	10,231,760	4,401,994	8036	1196
susan	30,198,836	16,983,309	3,040,300	6048	1058
qsort	469,467,608	71,431,808	57,562,752	4214	1084
FFT	140,521,122	23,813,697	17,374,787	3692	800
dijkstra	224,233,658	61,088,835	20,470,395	10,042	1068
patricia	609,908,006	115,437,095	86,679,508	15,042	1146
CRC32	61,659,073	6,965,007	12,292,816	230	1024
edn	79,170,256	15,731,129	13,997,301	1452	1600
nettle-aes	990,869,745	159,633,108	90,709,367	2880	10,566
lame	544,057,733	89,225,468	83,240,833	6074	8400
cholesky	12,506,634,754	1,813,527,857	760,514,353	18,078	30,800
lu	14,315,516,042	2,067,861,457	876,371,466	23,592	31,984

schemes. This is because the performance-aware cache management scheme can reduce the miss rate and allow the data in the cache to be optimally located by means of the data state and migration mechanism. Thus, even though the PACM scheme incurs larger access costs between the main memory and the NVM part of the L2 cache in the pre-backup stage, the overall performance is not degraded. In addition, the proposed algorithms together allow the communication overhead between memory and the NVM part of the L2 cache by virtue of the compression technique. Thus, the overall performance is the best among the three strategies. However, as the high voltage threshold increases, the overall performance advantage of the performance-aware cache management scheme decreases. For example, for JPEG, the performance gap between the dual-threshold approach and our proposed algorithm with a high voltage threshold of 1.14 V is 0.75×10^{11} cycle, but that with a high voltage threshold 1.35 V is only 0.15×10^{11} cycle. This is because the system can have more opportunities to wake from hibernation and thus avoid real power failures.

Table 5 shows that the execution time of the three strategies can be greatly reduced with thane increase in the high voltage threshold. For example, for JPEG, the execution time of our proposed algorithm with a high voltage threshold of



(a)



(b)

Fig. 3 The power pulse

1.14 V is 3.14×10^{11} cycles, and that with high voltage threshold of 1.35 V is 2.98×10^{11} cycles. However, for most benchmarks, when the high voltage threshold is increased to 1.35 V, the proposed algorithms require more run time compared with the high voltage threshold of 1.28 V. This occurs because the performance improvement decreases as the high voltage threshold increases, and the sleep time exceeds the maximum hibernation time when the high voltage threshold is 1.35 V. Since a sufficient time to wait for energy resumption cannot be guaranteed, we choose 1.28 V as the high threshold. According to the statistical comparison of a high voltage threshold of 1.28 V, the proposed algorithm and PACM can reduce the execution time by 17.5% and 3.8%, respectively, on average.

Since the RF traces do not harvest enough energy for most benchmarks, we also use solar traces to report the performance achieved. Figure 4 shows the performance comparison, in which the high voltage threshold is 1.28 V under solar energy traces. From the figure, it can be observed that the proposed algorithm and PACM outperform the dual-th algorithm for all benchmarks. On average, the proposed algorithm and PACM outperform dual-th by 13.5 and 4.9%, respectively.

Table 5 Performance evaluation ($\times 10^{11}$ cycles) with different high voltage thresholds

Bench.	Method	High voltage thresholds				Bench.	Method	High voltage thresholds			
		1.14 V	1.18 V	1.28 V	1.35 V			1.14 V	1.18 V	1.28 V	1.35 V
JPEG	Dual-th	3.89	3.68	3.12	3.13	qsort	Dual-th	40.56	38.43	37.43	35.28
	PAMC	3.39	3.18	3.08	3.09		PAMC	38.98	37.67	36.02	35.56
	Proposed	3.14	3.01	2.95	2.98		Proposed	33.27	30.73	29.67	29.7
susan	Dual-th	18.68	17.24	13.87	13.91	FFT	Dual-th	98.47	94.47	90.47	87.47
	PAMC	17.95	17.12	14.58	13.52		PAMC	93.17	90.17	88.17	87.17
	Proposed	13.89	13.03	12.07	12.15		Proposed	84.12	82.12	79.12	79.2
dijkstra	Dual-th	23.96	21.82	19.82	19.86	patricia	Dual-th	79.18	73.86	70.86	70.96
	PAMC	23.14	21.89	20.23	19.89		PAMC	78.16	73.97	72.13	72.15
	Proposed	18.23	17.23	16.23	16.36		Proposed	65.26	60.26	58.26	58.31
CRC32	Dual-th	4.31	4.20	4.15	4.15	edn	Dual-th	5.35	5.22	5.13	4.96
	PAMC	4.25	4.17	4.03	4.06		PAMC	5.30	5.19	5.04	4.96
	Proposed	4.19	4.10	4.03	4.06		Proposed	5.29	5.16	4.93	4.96
nettle-aes	Dual-th	114.9	95.75	87.84	85.40	lame	Dual-th	70.35	62.22	58.13	57.96
	PAMC	109.25	93.37	86.06	85.08		PAMC	67.30	57.27	52.54	51.61
	Proposed	106.85	91.32	83.18	81.95		Proposed	66.83	56.58	51.86	51.08
cholesky	Dual-th	271.82	248.75	229.64	234.40	lu	Dual-th	319.35	284.22	260.81	251.96
	PAMC	259.25	232.37	219.08	221.08		PAMC	305.53	275.26	252.54	247.58
	Proposed	261.85	226.91	208.75	213.83		Proposed	303.51	263.98	244.42	240.08

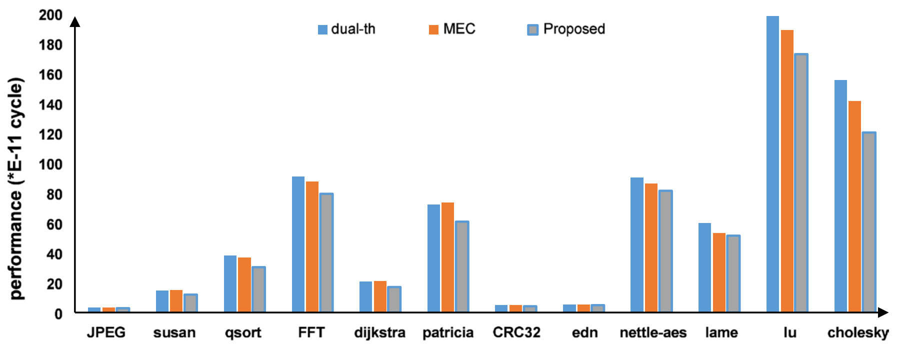


Fig. 4 Performance evaluation with a high voltage threshold of 1.28 V for solar energy traces

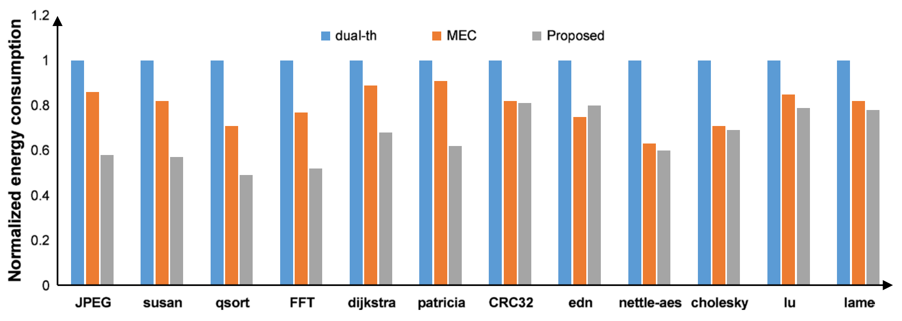


Fig. 5 Energy consumption normalized to that of the dual-threshold scheme with a high voltage threshold of 1.28 V for RF traces

5.2.2 Normalized energy consumption results

Figures 5 and 6 show the normalized energy consumption results obtained by running a set of simulations for all benchmarks based on the architectural model. The energy consumption consists of the scheduling energy consumption, the communication overhead, the switching overhead, and the resumption overhead. From the two figures, we know that the energy consumption of the proposed algorithms and that of the PACM algorithm are less than that of the dual-threshold scheme. Compared with the dual-threshold scheme, the PACM scheme achieves a 23.7% energy reduction on average. However, susan and patricia are different from other benchmarks, and the energy consumption of the PACM scheme is higher than that of the dual-threshold algorithm. This may be because the system reserves more energy for backup but has sufficient sleep time to wait for the system to recover. The dual-threshold scheme can reduce backup operations by more than 80% to avoid unnecessarily wasting energy, while the PACM scheme requires clearing the NVM part of the L2 cache, resulting in energy waste. In fact, there is no guaranteed sufficient time and energy to wait for the system to recover; inevitably, some backup operations cannot be avoided. As a result, the PACM scheme is superior to the dual-threshold scheme. Compared with

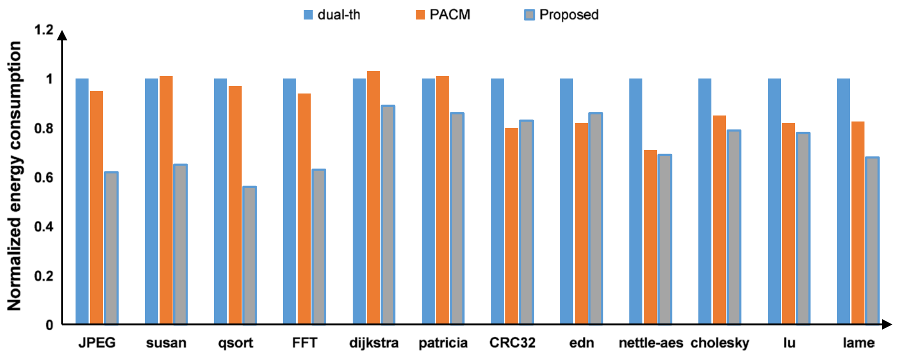


Fig. 6 Energy consumption normalized with a high voltage threshold of 1.28 V for solar traces

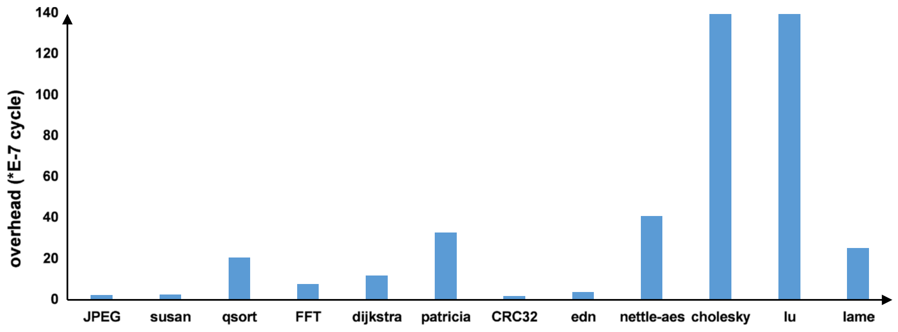


Fig. 7 Compression and decompression overhead for solar energy traces

the dual-threshold scheme, the proposed algorithms together can achieve a 52.6% energy reduction on average. For each benchmark, the proposed algorithms achieve an obvious energy reduction. This is because the use of compression technology can significantly reduce the communication overhead between the main memory and the processor.

5.3 Overhead

The overhead of the proposed algorithms includes two SRAM state bits and two L1 cache state bits for each block, the compression and decompression overheads for the NVM part of the L2 cache, and the data migration between volatile and nonvolatile blocks. With the system configuration shown in Table 2, the 4 kB L1 cache contains 64 blocks, and the 16 kB SRAM part of the L2 cache contains 256 blocks. Since there are two bits for each block, the overhead for the L1 cache and the SRAM part of the L2 cache is $(256 + 64) \times 2 = 640$ bits. The storage overhead is $64 \times 2/(4 \times 1024 \times 8) = 0.38\%$ for the L1 cache and $256 \times 2/(16 \times 1024 \times 8) = 0.39\%$ for the SRAM part of the L2 cache. The compression and decompression overheads depend on the benchmark size. Each NVM block needs to be compressed and decompressed at least once. Figure 7 shows the compression and decompression overhead. A comparison of Figures 4 and 4 shows that the performance overheads of compression and decompression are less than 1% of the total execution time. This is because the compression and decompression overheads are negligible compared to the reduction in the access cost between the NVM part of the L2 cache and the main memory. The results confirm the overall performance improvement even with these extra operations in the proposed algorithms.

6 Conclusion

Because frequent data backup and recovery operations significantly affect the performance and energy efficiency of NVPs, in this paper, we propose a performance-aware cache management scheme to improve performance and energy efficiency while guaranteeing successful backup. This scheme leverages data states and voltage thresholds for efficient cache management and data migration and prepares the system for backup when the voltage drops below the high voltage threshold, thus simultaneously maximizing execution progress and energy efficiency. To address the security and memory challenges presented by NVPs, this article also introduces an NVM compression technique based on the proposed performance-aware cache management scheme. The data in the NVM part of the L2 cache are compressed/decompressed depending on the benefits and penalties associated with cache hits and misses. The experimental results show that the proposed algorithms enable significant improvements in performance and energy efficiency.

Although our proposed algorithm can improve performance and reduce energy consumption, the need for nonvolatile processors and energy harvesters may be reduced as battery and power systems improve. Thus, we will study backup optimization and cache management issues in conventional systems in future work.

Acknowledgements This research was supported by the National Youth Program of the National Natural Science Foundation of China (Grant No. 61702123), and Science and Technology Program of Guangzhou, China (Grant No. 201804010461).

References

1. Embench: a modern embedded benchmark suite: [online]. https://lists.librecores.org/listinfo/embench
2. Mibench: [online]. http://www.eecs.umich.edu/mibench/
3. Measurement and instrumentation data center (MIDC) [online] (2020). http://www.nrel.gov/midc/
4. Albaseer A, Abdallah MM, Al-Fuqaha A, Erbad A (2021) Fine-grained data selection for improved energy efficiency of federated edge learning, Hamad Bin Khlifa University. arXiv:2106.12561
5. Binkert N, Beckmann B, Black G, Reinhardt SK, Saidi A, Basu A, Hestness J, Hower DR, Krishna T, Sardashti S et al (2011) The gem5 simulator. ACM SIGARCH Comput Arch News 39(2):1–7
6. Cronin P, Yang C, Zhou D, Qiu K, Shi X, Liu Y (2017) 'The danger of sleeping', an exploration of security in non-volatile processors. In: 2017 Asian Hardware Oriented Security and Trust Symposium (AsianHOST) 2017, pp 121–126

7. Fan W, Zhang Y, Song W, Zhao M, Shen Z, Jia Z (2020) Q-learning based backup for energy harvesting powered embedded systems. In: 2020 design, automation and test in Europe conference and exhibition (DATE). IEEE, pp 1247–1252
8. Habibzadeh M, Hassanalieragh M, Ishikawa A, Soyata T, Sharma G (2017) Hybrid solar-wind energy harvesting for embedded applications: supercapacitor-based system architectures and design tradeoffs. IEEE Circuits Syst Mag 17(4):29–63
9. Jalili M, Sarbazi-Azad H (2016) Endurance-aware security enhancement in non-volatile memories using compression and selective encryption. IEEE Trans Comput 66(7):1132–1144
10. Jog A, Mishra AK, Xu C, Xie Y, Narayanan V, Iyer R, Das CR (2012) Cache revive: architecting volatile STT-RAM caches for enhanced performance in CMPS. In: DAC Design Automation Conference 2012. IEEE, pp 243–252
11. Li J, Liu Y, Li H, Yuan Z, Fu C, Yue J, Feng X, Xue CJ, Hu J, Yang H (2018) Path: performance-aware task scheduling for energy-harvesting nonvolatile processors. IEEE Trans Very Large Scale Integr Syst 26(9):1671–1684
12. Li Q, Zhao M, Hu J, Liu Y, He Y, Xue CJ (2015) Compiler directed automatic stack trimming for efficient non-volatile processors. In: 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC). IEEE, pp 1–6
13. Li X, Ma K, George S, Khwa W-S, Sampson J, Gupta S, Liu Y, Chang M-F, Datta S, Narayanan V (2017) Design of nonvolatile SRAM with ferroelectric FETs for energy-efficient backup and restore. IEEE Trans Electron Dev 64(7):3037–3040
14. Liang X, Canal R, Wei G-Y, Brooks D (2007) Process variation tolerant 3t1d-based cache architectures. In: 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007). IEEE, pp 15–26
15. Liu J, Jaiyen B, Kim Y, Wilkerson C, Mutlu O (2013) An experimental study of data retention behavior in modern dram devices: Implications for retention time profiling mechanisms. ACM SIGARCH Comput Arch News 41(3):60–71
16. Liu Y, Yue J, Li H, Zhao Q, Zhao M, Xue CJ, Sun G, Chang M-F, Yang H (2017) Data backup optimization for nonvolatile SRAM in energy harvesting sensor nodes. IEEE Trans Comput Aided Des Integr Circuits Syst 36(10):1660–1673
17. Song W, Zhou Y, Zhao M, Ju L, Xue CJ, Jia Z (2018) EMC: energy-aware morphable cache design for non-volatile processors. IEEE Trans Comput 68(4):498–509
18. Sudevalayam S, Kulkarni P (2010) Energy harvesting sensor nodes: survey and implications. IEEE Commun Surv Tutor 13(3):443–461
19. Wang Y, Jia H, Liu Y, Xue CJ, Yang H, et al (2014) Register allocation for hybrid register architecture in nonvolatile processors. In: 2014 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, pp 1050–1053
20. Wang Y, Liu J, Hu J (2020) Communication-aware task scheduling for energy-harvesting nonvolatile processors. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2020, 28(8):1796-1806
21. Zhao M, Fu C, Li Z, Li Q, Xie M, Liu Y, Hu J, Jia Z, Xue CJ (2017) Stack-size sensitive on-chip memory backup for self-powered nonvolatile processors. IEEE Trans Comput Aided Des Integr Circuits Syst 36(11):1804–1816
22. Zhao M, Li Q, Xie M, Liu Y, Hu J, Xue CJ (2015) Software assisted non-volatile register reduction for energy harvesting based cyber-physical system. In: 2015 Design, Automation and Test in Europe Conference and Exhibition (DATE). IEEE, pp 567–572
23. Zhao M, Qiu K, Xie Y, Hu J, Xue CJ (2016) Redesigning software and systems for non-volatile processors on self-powered devices. In: 2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC). IEEE, pp 1–6
24. Zhou D, Qiu K, Xu Y, Shi X, Liu Y (2018) A dual-threshold scheme along with security reinforcement for energy efficient nonvolatile processors. In: 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, pp 70–75
25. Zhou Y, Zhao M, Ju L, Xue CJ, Li X, Jia Z (2017) Energy-aware morphable cache management for self-powered non-volatile processors. In: 2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). IEEE, pp 1–7
26. Zhu M, Pham H (2020) An empirical study of factor identification in smart health-monitoring wearable device. IEEE Trans Comput Soc Syst 7(2):404–416

Authors and Affiliations

Yan Wang[1] · **Kenli Li**[2] · **Xia Deng**[1] · **Keqin Li**[2,3]

[1] School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510008, China

[2] College of Information Science and Engineering, Hunan University, Changsha 410082, China

[3] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA