electronics

MDPI

Review

Crash Recovery Techniques for Flash Storage Devices Leveraging Flash Translation Layer: A Review

Abdulhadi Alahmadi [1] and Tae Sun Chung [2,*]

[1] Department of Computer Engineering, Ajou University, Suwon 16499, Republic of Korea
[2] Department of Artificial Intelligence, Ajou University, Suwon 16499, Republic of Korea
* Correspondence: tschung@ajou.ac.kr

Abstract: The flash storage is a type of nonvolatile semiconductor device that is operated continuously and has been substituting the hard disk or secondary memory in several storage markets, such as PC/laptop computers, mobile devices, and is also used as an enterprise server. Moreover, it offers a number of benefits, including compact size, low power consumption, quick access, easy mobility, heat dissipation, shock tolerance, data preservation during a power outage, and random access. Different embedded system products, including digital cameras, smartphones, personal digital assistants (PDA), along with sensor devices, are currently integrating flash memory. However, as flash memory requires unique capabilities such as "erase before write" as well as "wear-leveling", a FTL (flash translation layer) is added to the software layer. The FTL software module overcomes the problem of performance that arises from the erase before write operation and wear-leveling, i.e., flash memory does not allow for an in-place update, and therefore a block must be erased prior to overwriting upon the present data. In the meantime, flash storage devices face challenges of failure and thus they must be able to recover metadata (as well as address mapping information), including data after a crash. The FTL layer is responsible for and intended for use in crash recovery. Although the power-off recovery technique is essential for portable devices, most FTL algorithms do not take this into account. In this paper, we review various schemes of crash recovery leveraging FTL for flash storage devices. We illustrate the classification of the FTL algorithms. Moreover, we also discuss the various metrics and parameters evaluated for comparison with other approaches by each scheme, along with the flash type. In addition, we made an analysis of the FTL schemes. We also describe meaningful considerations which play a critical role in the design development for power-off recovery employing FTL.

Keywords: storage management; software defined module; power failure; NAND; FTL; flash memory; crash recovery

Citation: Alahmadi, A.; Chung, T.S. Crash Recovery Techniques for Flash Storage Devices Leveraging Flash Translation Layer: A Review. *Electronics* **2023**, *12*, 1422. https://doi.org/10.3390/electronics12061422

Academic Editor: Shinichi Yamagiwa

Received: 16 February 2023
Revised: 8 March 2023
Accepted: 14 March 2023
Published: 16 March 2023

1. Introduction

Flash storage devices are electronic devices that use nonvolatile flash memory to store data. They are often used as an alternative to traditional hard disk drives (HDDs) because they are more durable, lighter, and faster. There are several types of flash storage devices, including:

- USB flash drives: small, portable devices that can be plugged into a computer's USB port to transfer data.
- Solid-state drives (SSDs): larger storage devices which are used as an alternative to traditional HDDs in computers and servers.
- Memory cards: small, portable devices that are used to store data in electronic devices such as cameras, smartphones, and tablets.
- External hard drives: larger storage devices that can be connected to a computer via a USB or other interface to store and transfer data.

Flash storage devices have several advantages over traditional HDDs. They are more durable because they have no moving parts, which makes them resistant to physical shock and vibration. They are also faster because data can be accessed directly from the memory, rather than having to be retrieved from a spinning disk. In addition, flash storage devices are generally smaller, as well as lighter than HDDs, which makes them more portable [1–4].

Solid-state drives (SSD) [5–7] and mobile phones frequently employ flash memory because of its nonvolatility, low power usage, quick access speed, and shock resilience. Solid-state drives, which utilize NAND flash memory, are now on the rise and are even competing with hard drive sales [8]. Database servers can operate more quickly thanks to flash-based storage devices, which are thought to be a novel storage method that can replace disks [9]. The unit of operations in flash memory is different, hence the overwrite operation cannot be carried out directly. At the block level, which consists of numerous pages that can meet up to 1.5 ms, an erase operation is carried out. In contrast, a page is the unit, where read and write operations are carried out, and it may match up to 80 µs and 200 µs incessantly, respectively [10]. However, NAND flash memory features a hardware function known as the operation of erase before write (out of place update) that causes a page to be wiped before being written in the same spot [11]. An intermediary software layer, which is called the flash translation layer (FTL), was established, while the memory part for erasing differs in size from the section for reading or writing [12]. Figure 1 [13] reveals how demand for flash is powered mainly by various major markets—mobile phones, solid-state drives, flash memory cards, MP3/PMP players, and flash drives.

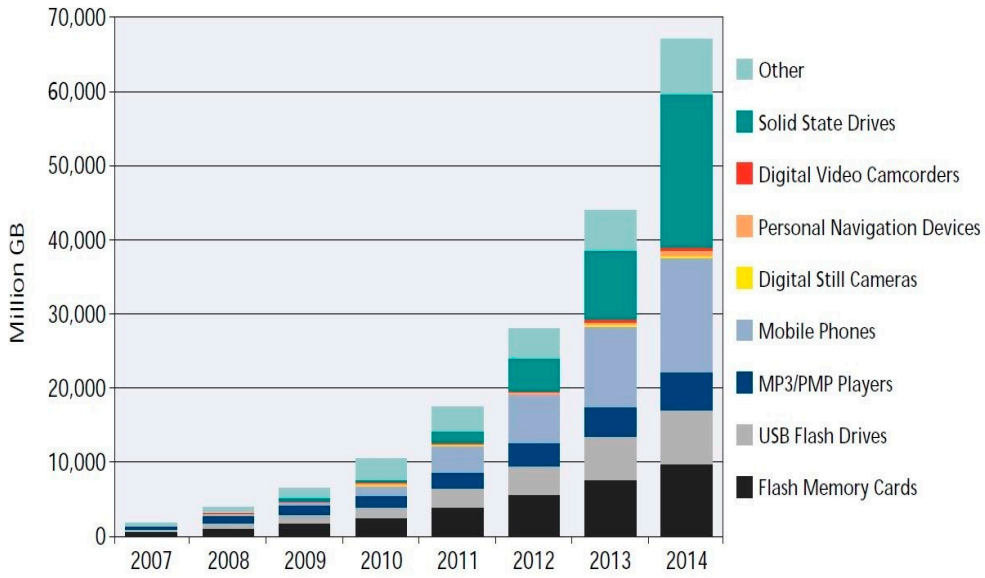

Figure 1. The driving of flash memory by different markets [13].

Numerous FTLs have been proposed as the FTL has a significant impact on the performance features of flash memory devices [14–21]. The effectiveness of FTLs is particularly significant when considering the ever-increasing capability of flash memory-related devices. According to an address mapping approach, the present FTLs may be broadly divided into page, block, and hybrid mapping [12]. The FTL is primarily accountable for address mapping, wear-leveling, and trash collection.

The majority of FTL research to date has, however, mostly concentrated on performance and wear-leveling difficulties. The recovery regarding a power-off failure in the field of FTLs, however, has not received much attention despite its practical relevance. In particular, there are not many books that thoroughly discuss crash recovery in FTLs. In addition, a lot of flash-based solid-state drive (SSD) suppliers add their particular crash recovery schemes and algorithms; however, none of them have been issued and published because the vendors are hesitant to share their solutions. The tightly connected crash

recovery algorithm in conjunction with FTL, in our opinion, has a lot of space for growth and shows promise for meeting both the needs of efficient crash recovery and low overhead throughout normal execution times. Hence, in this paper, we review several crash recovery approaches. The main contributions of our paper are as follows:

1. First, we explain the preliminary knowledge of flash storage devices considering the crash recovery process.
2. Then, we provide a taxonomy of the FTL algorithms as the crash recovery in the paper leverages the FTL.
3. We comprehensively illustrate the schemes that address crash recovery leveraging the FTL.
4. The comparative overview for the mapping methodology is demonstrated in Table 1.
5. We also review the recovery process in state of the art machine learning-based schemes for various flash devices; the flash types with parameters are described in Tables 2 and 3.
6. Moreover, we discuss the various approaches, and the classification regarding their mapping methodology are shown in Table 4.
7. Finally, we present significant considerations that are critical in the design process for power-off recovery, utilizing FTL.

Table 1. The comparative overview of mapping in the FTL algorithms.

Scheme	Cost of Reading	Computational Overhead	Size of the Mapping Table
Sector mapping	Small	No	Greater
Block mapping	Small	Yes	Smaller, as compared with sector mapping
Hybrid mapping	Large	Yes	Equal to block mapping

The rest of the paper is arranged as follows: Section 2 discusses the background knowledge for crash recovery using flash storage. In Section 3, we explain the taxonomy of the FTL algorithms. In Section 4, we illustrate the motivation of using FTL. In Section 5, we discuss various approaches for crash recovery, along with their parameters of comparison and flash types. We analyze and compare different approaches using illustrations and tables. The conclusion of the paper is given in Section 6.

2. Preliminaries

In this section, we discuss the preliminary topics of our review study to give an insight to readers on the background knowledge of the FTL and flash memory types, such as NAND, SSDs, flash drives, and HDD, along with flash memory features for operations.

2.1. NAND Flash Memory

The NAND flash is a category of nonvolatile memory technology that is commonly used in electronic devices. It is named after the NAND gate, which is a type of electronic gate that performs a NOT–AND operation. The NAND flash stores data using an array of floating-gate transistors. It has several advantages over traditional hard disk drives (HDDs). It is more durable because it has no moving parts, which makes it resistant to physical shock and vibration. It is also faster because data can be accessed directly from the memory, rather than having to be retrieved from a spinning disk. In addition, it is generally smaller and lighter than HDDs, which makes it more portable. Moreover, NAND is used in a wide range of electronic devices, including USB flash drives, mobile phones, solid-state drives (SSDs), and memory cards. It is also used in some external hard drives and in some computer systems as a replacement for traditional HDDs.

The NAND flash storage is comprised of planes, which are made of various dies. As demonstrated in Figure 2 [13], every die comprises several blocks. Moreover, at this phase, erase-related operations are accomplished. Surrounded by a block, there are various pages, that are basically the coarseness at which operations such as read and write are fulfilled. In addition, a page further is subdivided into a small spare area, and a data area (which is typically around 1/32 of the size of data) known as out of band (OOB). Moreover, the OOB for a page is generally utilized for storing the information of the page state, as well as the logical page number (LPN) for a page. Moreover, an OOB also encompasses an error correction code (ECC), which is used for checking the accuracy of the stored data on the data of a page area, as indicated in Figure 2. Throughout a read process, the whole page is to be read simultaneously, as well as the area of OOB, even though a write operation can be performed selectively on both the data area or OBB.

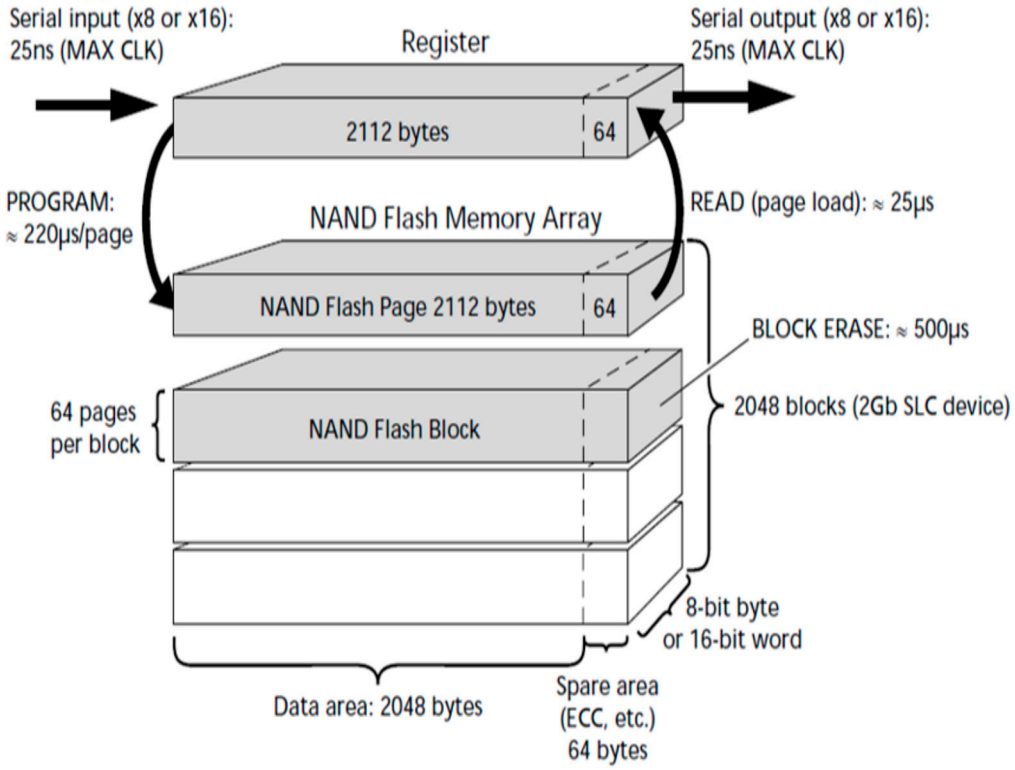

Figure 2. The organizational overview and architecture of NAND flash. (Source is: Micron Technology Inc.) [13].

2.2. SSDs

The solid-state drive (SSD) is a kind of storage device which utilizes NAND-based flash memory for the purpose of storing data. In contrast to a conventional hard disk drive (HDD), which stores data on rapidly spinning disks, an SSD has no moving parts and relies on electronic circuits to store and retrieve data. This makes SSDs faster, more reliable, and more durable than HDDs. Further, SSDs are commonly used in laptops, desktop computers, and servers, to store the operating system, applications, and data. They are also used in some mobile devices, such as smartphones and tablets.

2.3. Flash Drive

A flash drive, also known as a USB drive or a thumb drive, is a portable storage device that uses NAND-based flash memory in order to store data. It is small, lightweight, and easy to use, making it a convenient way to transfer files between computers and other devices. Flash drives are typically plugged into a computer's USB port and can be used to store and transfer a large variety of files, including documents, photos, videos, and

music. They are also commonly used to store and transfer data between devices, such as smartphones, tablets, and cameras. Flash drives are available in a range of capacities, with some models offering up to 1 terabyte of storage.

2.4. HDD

The HDD is a form of storage that uses rapidly spinning disks for storing, as well as retrieving, data. It is the common type of storage device found in computers, servers, and other electronic devices. An HDD stores data magnetically on spinning disks called platters. A read/write head mounted on an armature moves over the spinning disks to access the data. Of note, HDDs are slower than SSDs, and they leverage NAND-based flash memory to store data; however, they are generally less expensive and offer larger storage capacities. Additionally, HDDs are commonly used to store the operating system, applications, and data in computers and servers, as well as to store large volumes of data in external storage devices.

Figure 3 demonstrates that the local file systems (native) are utilized directly upon flash memory structures devoid of support for block-device simulation. Because this article concentrates on FTL, the subsequent discussion presents FTL.

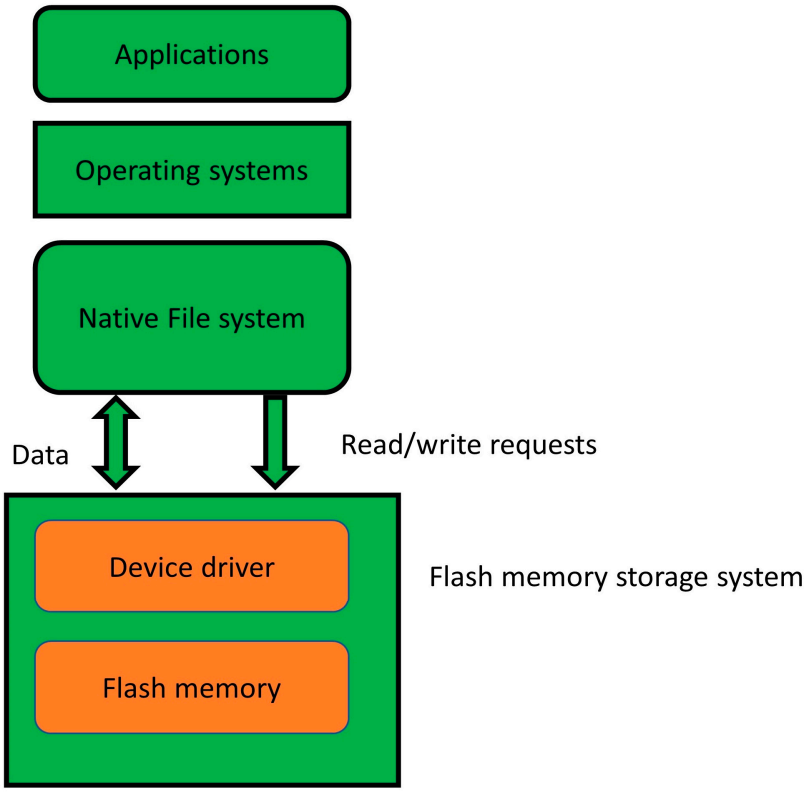

Figure 3. Features of a flash memory management system.

2.5. Flash Translation Layer

A FTL is a software layer which is employed to oversee the storage and retrieval of data in a flash storage device, such as an SSD or a flash-based USB drive. In a flash-based device, data is stored on flash memory chips, which are nonvolatile, and can maintain data even when the power for the corresponding device is off. However, flash memory has some limitations, which make it more complex to manage compared to traditional magnetic storage media, such as hard drives. As an example, cells of the flash memory can only be written up to a controlled quantity of times before they wear out, and they can become slower as they fill up with data.

The FTL is responsible for managing these limitations and ensuring that the flash-based storage device performs optimally. It performs tasks such as wear-leveling, which spreads the writes evenly across the flash memory chips to extend their lifespan, and garbage collection, which reclaims space that has been freed-up by deleting files. The FTL also manages the mapping for logical block addresses (LBAs), used by the operating system to the physical locations [22–24] on the memory chips of flash, where the data is stored. The FTL is a valuable component of a flash-based storage device and plays a decisive role in maintaining its performance and reliability. It is typically implemented in firmware and is transparent to the operating system and the user. Figure 4 illustrates the application, FTL, flash memory, and file system logical flow. The idea revolves around the FTL layer. Some advantages of the FTL include:

1. Improved performance: the FTL can recover the performance for a SSD by managing the physical position of data on the device and by optimizing the order in which data is written to the device.
2. Wear-leveling: the FTL assists to broaden the life for a SSD by distributing writes evenly across the entire device, a technique known as wear-leveling. This helps to prevent any single location on the device from being worn out due to excessive writing.
3. Bad block management: the FTL can also manage bad blocks on the device, mapping them out and substituting good blocks in their place.
4. Compatibility: the FTL enables a SSD to be used as a drop-in replacement for a traditional hard disk drive (HDD) by presenting itself to the host system as an HDD. This allows a SSD to be used in systems that may not have been designed specifically for solid-state storage.

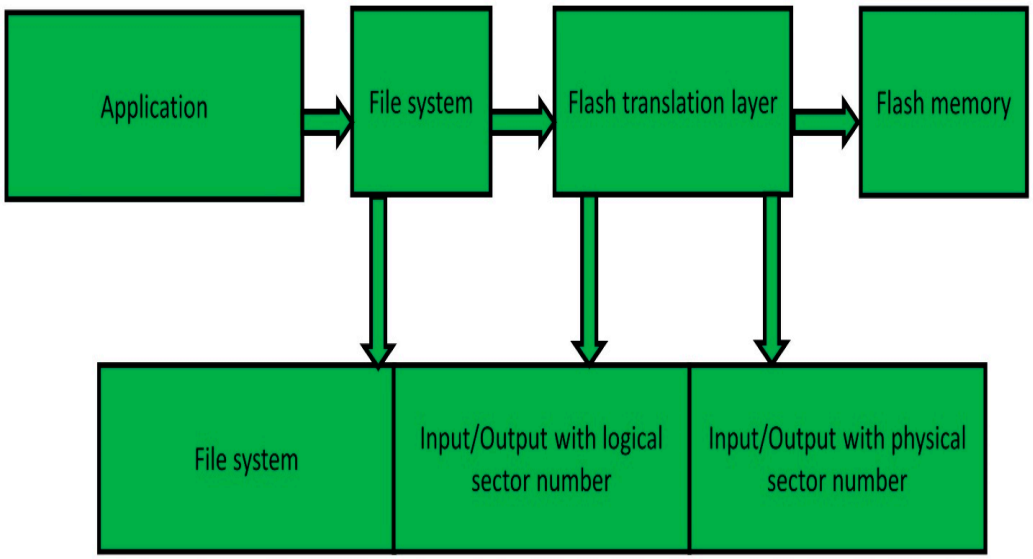

Figure 4. An architectural overview of a flash memory system.

2.6. Flash Memory Operations Characteristics

The read/write actions in flash are asymmetric in terms of processing speed. Since they are carried out at a coarser roughness for block, erase procedures are thought to be significantly slower compared to read, as well as write functions [25]. The access period for reads and writes is also asymmetrical. Additionally, the write delay that occurs during erase is typically greater from read latency (via a 4–5 factor). The writing process takes a longer time than reading as it takes longer to remove electrons from a flash cell, which is the main rationale for this. In a flash memory, the out of place updates are considered as the most economical; this is because the writes happen page by page, whereas erase operations happen at the block granularity level. An inappropriate update renders the

current edition of the page invalid when the host releases an upgrade. The mapping table is then altered, and the amended new translation is written into a fresh flash file. Daily updates are required for mapping tables. The FTL keeps track of these modifications. There are a lot of faulty pages as a result of flash memory's out of place upgrading capabilities. Null pages are those that have outdated content that has to be removed. Garbage disposal reclaims these empty pages by removing the blocks that contain them. All pages within the block must have faulty data in order to remove a lock.

3. The Taxonomy of the FTL Algorithm

Herein, we describe the taxonomy for the FTL algorithm corresponding to characteristics that encompass address mapping, and mapping information organization. The authors in [26–29] provide ideas for the classification of FTL algorithms. However, herein, we provide a detailed description of this classification in the FTL.

3.1. Sector Mapping Approach

Sector mapping refers to the process of mapping logical sectors (also known as logical blocks) of the file system to physical sectors (also known as physical blocks) on the flash memory. This mapping is maintained in a table, often referred to as the block map or page map.

The purpose of sector mapping is to provide a level of abstraction between flash memory and the file system, allowing the file structure to treat the flash storage as if it were a conventional rotating disk. The FTL is responsible for managing the physical characteristics of the flash memory, for example, wear-leveling, serious block management, and garbage collection.

Sector mapping is performed dynamically by the FTL, and the mapping can change as new data is recorded to the flash memory and old data deleted. The FTL must ensure that the mapping is consistent, and that data are not missing in the event of a power failure. This requires the use of write buffering, metadata management, and proper error handling to prevent data corruption or loss.

Sector mapping can also be used to implement various performance optimization techniques, such as wear-leveling, which spreads writes evenly across the flash memory to ensure that all blocks have an equal number of writes, and garbage collection, which reclaims blocks that are no longer in use and makes them available for new data.

In summary, sector mapping is an essential component of a FTL, allowing it to provide reliable and efficient access to flash memory while hiding the physical attributes of the flash from the file system.

3.2. Block Mapping Strategy

Block mapping in the FTL is the process of mapping logical blocks for a file system to the physical blocks upon the flash. In addition, the mapping is stored in a table, often referred to as the block map or page map, which is maintained by the FTL.

Block mapping's goal is to provide an abstraction level in the middle of the file system and the flash memory, so that the file system may treat the flash memory like a regular spinning disk. The FTL is in charge of controlling the flash memory's physical properties, including wear-leveling, faulty block management, and trash collection.

The FTL dynamically performs block mapping, and when new data are written to the flash memory and old data removed, the mapping may change. The mapping must be constant, and the FTL must make sure that data does not get wasted in the case of a power outage. To prevent data loss, this necessitates the use of write buffering, metadata management, and appropriate error handling.

Block mapping can also be used to implement various performance optimization techniques, such as wear-leveling, which spreads writes evenly across the flash memory to ensure that all blocks have an equal number of writes, and garbage collection, which reclaims blocks that are no longer in use and makes them available for new data.

In summary, block mapping is an essential component of the FTL, allowing it to provide reliable and efficient access to flash memory, while hiding the physical traits of the flash-based memory from the file system.

The review of the sector, block, and hybrid mapping with respect to the cost of reading, and the computational overhead of the size of the mapping table, with respect to each mapping methodology, is given in Table 1.

3.3. Hybrid Mapping Methodology

Hybrid mapping in a FTL is a method of mapping logical blocks of the file system to physical blocks on the flash memory, which combines elements of both page mapping and block mapping.

In page mapping, every logical page can be mapped to a single physical page, while in block mapping, every logical block is mapped in corresponding with a single physical block. In hybrid mapping, a logical block is divided into multiple pages, and each page is mapped to a physical page.

Hybrid mapping is used to provide a balance between the benefits of both page mapping and block mapping. The small size of pages in page mapping allows for more efficient wear-leveling and garbage collection, while the larger size of blocks in block mapping provides better performance and reduces overhead.

With hybrid mapping, the FTL can maintain the benefits of both page mapping and block mapping, while also allowing for more fine-grained control over the mapping process. This can result in improved performance and reliability compared to using either page mapping or block mapping alone.

In addition, hybrid mapping allows for the use of various flash management algorithms, such as wear-leveling, bad block management, and garbage collection, to optimize the performance and reliability of the flash memory.

In summary, hybrid mapping is a flexible and efficient method of mapping logical blocks to physical blocks in a flash translation layer, providing the benefits of both page mapping and block mapping.

3.4. Log Block Method

A log block within a FTL is a block of the flash memory that is used to store data temporarily before being written to a final location on the flash memory. The log block is used to improve the write performance of the flash memory by allowing data to be written to the log block and then later transferred to their final location in a single operation.

The use of log blocks in a FTL is known as log-structured flash memory (LFSM) [26–29] and is designed to prevail over the shortcomings of flash memory in terms of its write performance and endurance. The log block acts as a buffer for incoming write requests, allowing multiple write requests to be combined into a single, larger write operation. This can result in improved write performance and reduced write amplification, which is a measure of the amount of data written on the top of flash memory relative to the amount of data requested to be written.

Moreover, log blocks can be used to implement various flash management algorithms, such as wear-leveling and garbage collection, which can improve the performance and reliability of the flash memory.

In short, log blocks are an important component of a flash translation layer and are used to improve the write performance and reliability of flash memory. They provide a flexible and efficient way to manage the write requests to the flash memory and ensure that data are written quickly and reliably.

3.4.1. BAST

The block associative sector translation (BAST) is a method of flash translation in which a group of logical sectors is mapped to a single physical block of the flash memory.

In BAST, a block of flash memory is divided into a number of pages, and each page is associated with a single logical sector. The mapping information is stored in a table in RAM, which is used to translate the logical addresses to physical addresses. The mapping information is updated dynamically as data are written to and deleted from the flash memory.

The BAST is used to implement the FTL for flash memory-based storage devices. The advantage of BAST is that it provides better performance compared to other methods of flash translation, as each physical block can contain multiple logical sectors, reducing the number of erase operations required to write data to the flash memory.

In comparison to other methods of flash translation, such as page mapping and fully associative sector translation (FAST), the BAST provides a balance between the flexibility of the mapping process and the performance corresponding to flash memory.

However, the BAST also has a higher overhead compared to page mapping, as the mapping information must be stored in RAM, and the mapping table must be updated dynamically as data are written to and deleted from the flash memory.

In summary, the BAST is a method of flash translation that provides a balance between the flexibility of the mapping process and the performance of the flash memory. It is used in flash translation layers to provide efficient and reliable access to flash memory-based storage devices.

3.4.2. FAST

The fully associative sector translation (FAST) [26–29] is a method of flash translation in which every logical sector of the file system can be mapped to any physical sector of the flash memory. In addition, the FAST is associated as a part of the FTL covered in various schemes and methodologies.

In FAST, the mapping information is stored in a table in RAM, which is used to translate the logical addresses to physical addresses. It divides flash storage into two types of areas, i.e., data and log. The mapping information is updated dynamically as data are written to and deleted from the flash memory.

Moreover, the FAST is utilized to execute the FTL for storage devices leveraging flash memory. The advantage of the FAST is that it provides more flexibility in the mapping process, as every logical sector has the capability of mapping it to every physical sector, allowing for efficient wear-leveling, as well as garbage collection algorithms.

In comparison to other methods of flash translation, such as block mapping and page mapping, the FAST provides the greatest degree of freedom in mapping logical sectors to physical sectors, which can result in the improved performance and reliability of the flash memory.

However, the FAST also has a higher overhead compared to other methods of flash translation, as the mapping information must be stored in RAM, and the mapping table must be updated dynamically as data are written to and deleted from the flash memory.

In brief, the FAST is a method of flash translation that provides a high degree of flexibility in the mapping process, but also has a higher overhead compared to other methods of flash translation. It is used in flash translation layers to provide efficient and reliable access to flash memory-based storage devices.

4. Motivation for Using FTL in Crash Recovery

In crash recovery scenarios, the FTL can play a crucial role in ensuring the integrity of the stored data. When a system experiences a crash, it may result in incomplete or corrupted data being written to the flash storage. In such cases, the FTL can help in recovering the data by mapping the logical addresses to their corresponding physical addresses on the flash storage. The FTL can also reduce the time required for crash recovery. Without the FTL, the system would need to scan the entire flash storage to locate and recover the data. This can be a time consuming process, especially when dealing with a large amount of

data. With the FTL, the system can quickly locate and recover data by mapping the logical addresses to their corresponding physical addresses.

In addition, the FTL can also help in reducing the frequency of crashes by using techniques such as wear-leveling and garbage collection to ensure that the flash storage is used efficiently and effectively. This can help to prevent the premature wear and failure of the flash storage, which can lead to crashes. In summary, the motivation for using the FTL in crash recovery is to ensure the integrity of the stored data, reduce the time required for recovery, and prevent premature wear and failure of the flash storage.

Moreover, the FTL is responsible for translating logical block addresses to physical block addresses on the underlying NAND flash memory. Besides its primary role in SSDs, the FTL can also be leveraged to provide crash recovery mechanisms that can help prevent data loss and corruption in the event of system crashes. Herein, we illustrate various benefits of using the FTL for crash recovery. By using the FTL to manage the mapping of logical to physical block addresses, the SSDs can implement crash recovery mechanisms that help ensure data integrity and prevent data loss in the event of a system crash. This can help improve the overall reliability of the storage system, which is especially important in applications that require a high level of data availability. The FTL-based crash recovery mechanisms can help reduce the recovery time in the event of a system crash, compared to traditional storage systems. This is because the FTL can maintain a mapping of logical to physical block addresses in nonvolatile memory, which allows it to quickly recover the mapping and resume normal operation following a crash [12–14].

Similarly, the FTL-based crash recovery mechanisms can also help reduce maintenance costs, as they can eliminate the need for data backup and other data recovery processes that may be required in traditional storage systems. This can help simplify the storage management process and reduce the overall cost of ownership for the storage system. The FTL-based crash recovery mechanisms can also help improve overall system performance, as they can eliminate the need for time consuming data recovery processes that may be required in traditional storage systems. This can help improve application performance and reduce system downtime, which can have significant benefits in applications where high availability is critical.

Ultimately, the FTL-based crash recovery mechanisms can provide several advantages over traditional storage systems, including improved reliability, faster recovery time, lower maintenance costs, and improved performance. By leveraging the advanced features of modern SSDs, such as the FTL, organizations can help ensure that their critical data remain available and secure, even in the event of system crashes.

5. The FTL-Based Crash Recovery Schemes and Discussion

Due to the practical significance of crash recovery methods in a FTL, we herein discuss various state of the art works that deal with the crash recovery problem in the FTL. First, we explain the schemes that utilize machine learning with the FTL for crash recovery. Moreover, in Tables 2 and 3 we provide a summary of the schemes, with respect to their flash-type simulators and metrics of comparison for the NAND and SSD flash types. Further, Table 4 provides a summary of mapping strategies, i.e., sector, block, as well as hybrid.

5.1. Machine Learning-Based Methods for Crash Recovery

The authors proposed an approach based on machine learning for the error recovery of low-delay NAND flash devices. In general, the maintenance time and activity variation of a NAND flash controller are obscure. Hence, the selection of a suitable decoding mode is challenging for a controller, causing an increase in the latency. To substitute the inaccessible data, the authors employed the on-cell ratio as alternative data. The authors' suggested error recovery strategy selects an appropriate decoding mode that guarantees effective error recovery along with a minimum latency for an average reading operation. Through simulation, they demonstrated that their proposed approach predicts the best possible

decoding mode together with high-level accuracy. Furthermore, the performance of average latency for a read is substantially improved, as compared with other baselines [30,31].

The first finding of this study was that groups of shared pages should be studied independently, since they have distinct raw bit error rate (RBER) values. Utilizing the characterization data of a brand new, unused flash memory chip, we built a machine learning model using this realization to forecast the blocks that would have reduced durability (s). To do this, location-sensitive and value-sensitive features from the shared pages group were extracted and engineered into more complex and comprehensible characteristics. We also discussed how lifespan may be increased by combining our suggested prediction model with the current FTL wear-leveling mechanism. The support vector machine (SVM) outperformed the other machine learning algorithms we assessed for the proposed prediction and longevity enhancement method [32], achieving higher accuracy of up to 85% at a reduced computational cost.

In this paper [33], the authors utilized a dynamic ERF leveraging machine learning (ML), which predicts an ideal decoding mode, ensuring productive decoding and the least latency of read operation and uses it directly for a reduction of the read latency. Owing to process differences experienced in the memory manufacturing procedure, the characteristics of memory are different via chips; therefore, it becomes difficult to utilize a trained model of prediction on different chips. Moreover, the training of the customized model (of prediction) at every chip of memory is infeasible since the computational overhead for training is severe, and a huge amount of training data were used. Therefore, the authors leverage ERF prediction considering reusable ML, which deals with changing input as well as output relationships through chips, because of process variation. The ML strategy reuses a pretrained model design, or knowledge acquired, as source tasks for adjusting the model towards performing its task with no performance loss in various chips. The authors adopt two reusable ML methods for the prediction of the ERF based upon metalearning and transfer learning. The transfer learning strategy reuses the model, which is pretrained via reducing domain change between the source and target chips, utilizing an algorithm for domain adaptation. In contrast, the metalearning approach learns common features from several source chips throughout the procedure of metatraining. Then, the trained model (meta) reuses formerly learned knowledge towards quickly adapting, according to various chips. The numerical results show the benefits of the proposed scheme through the high accuracy of prediction in several chips. Moreover, the suggested ERF prediction, which is based upon transfer learning and metalearning, results in a reduction of read latency in comparison with conventional strategies.

5.2. Various Flash Types Leveraging FTL in Crash Recovery and Parameters of Evaluation

In study [34], the authors have proposed two crash recovery schemes that need fewer write operations on flash memory. The suggested methods remove redundant flash writes via storing metadata simultaneously, through employing a spare area linked with every data page.

In flash memory, the volatile memory is utilized for the storage of frequently accessed mapping tables. Nevertheless, the tables can be lost upon a sudden failure of power. Hence, the authors introduce a recovery scheme with optimized metadata that retrieves the tables more quickly than other state of the art methods. A compression methodology is utilized, where several data along with their metadata are stored on a physical page through utilizing a fast strategy for the incompressible and compressible data. In the strategy of incompressible data, the storage of metadata are on the physical pages of compressible data. In contrast to this, in the method for compressible data, to recover the data, the metadata for the preceding pages are embedded in the data area for every physical page. This method has lower area operating costs with higher performance, as compared with related works. In addition, the results from the simulation reveal that the proposed approach with real workloads surpasses other schemes through a significant decrease in the time of recovery i.e., up to 92.42% [35].

As a storage medium, the NAND flash memory has gained widespread acceptance. Crash recovery turns out to be crucial in NAND flash memory storage procedures since power outages can happen several times, and cause data loss. The flash translation layers (FTLs) are needed to control flash memory; therefore, how to properly and consistently recover the FTL metadata after a system crash is the crux of the crash recovery challenge in the NAND flash. Moreover, the deterministic crash recovery (DCR), which uses a deterministic technique for crash recovery in the NAND flash storage devices, is presented by the authors in [36]. The core concept is to use crash recovery to take advantage of the determinism of the FTLs and recreate the events that took place between the past checkpoint and the current crash point. By merely examining a small number of blocks dependent on deterministic FTL processes, the DCR can restore the system more effectively than previous methods that require scanning the whole flash memory chip. We put the DCR into practice in a FTL and contrasted it with typical version-based and power loss recovery strategies based on embedded ARM systems. According to experimental findings, the DCR can significantly shorten recovery times and ensure the consistency of the FTL information following recovery.

A unique crash recovery plan is offered by the authors of [37] for FAST, a hybrid address mapping FTL. It uses the FAST FTL property that log blocks in a log area and are utilized in a round-robin fashion to write regularly freshly created address mapping information in a log-structured manner, giving it two benefits over the current FTL recovery systems: one is the low overhead involved in logging throughout regular FTL operations; the other is the quick recuperation time.

Table 2. Various methods, flash types, and performance parameters for comparison of the NAND.

Schemes and Strategies Used in Literature	Type of Flash	Parameters Used for Evaluation and Comparison
Redundancy removing [34]	NAND	Reduce write operations upon the flash memory
DCR [36]	NAND	Reduction in recovery time
FAST FTL [37,38]	NAND	Recovery time evaluation
Fast Check [39]	NAND	Logging overhead, recovery time
PORCE [40]	NAND	Time of recovery
DSFTL [41]	NAND	Overhead associated with trash operations, erase operations reduction, increase of switch merge operations
FTRM [42–45]	NAND	To check the reliability of the recovery scheme
Timing analysis [46]	NAND	Time of crash recovery
Logging method [47]	NAND	Power of recovery, metadata overhead
Legacy logging scheme [48]	NAND	Cost of recovery, page mapping reliability
Lazy FTL [49]	NAND	Scalability, response latency, and overhead for merging operations
RSLSP [50–52]	NAND	Recovery latency, comparison of latency with recovery and without recovery
Page mapping [53]	NAND	Recovery time and management costs for mapping information

Flash memory has expanded in popularity in recent years as a result of its benefits, including quick data access, low power consumption, and great mobility. However, there are other issues with flash memory that need to be resolved, such block deletion restrictions and erase before write issues. The FTL (flash translation layer) has been presented as a solution to this problem, and includes helpful capabilities including wear-leveling, trash collection, and address mapping. In the event that the storage systems lose power while being used, data may be lost. The data might be vital in some systems. Data recovery

is therefore crucial in the case of a system disaster or a sudden power interruption. Researchers have been interested in this issue, and several studies have been conducted. In this work, we examine early research on data recovery for flash memory, ranging from FTL processing techniques to PLR (power loss recovery) explanations that have been suggested by authors in conference proceedings, patents, or in scholarly magazines. This will include an overview and a review of the solutions that have been suggested for data recovery in flash memory [38].

Table 3. Various techniques, flash type, and performance metrics for comparison of the SSD.

Schemes and Strategies Used in Literature	Flash Type Simulator for Performance Analysis	Metric/Parameter for Performance Analysis and Comparison
Optimized metadata [35]	SSD	Recovery time reduction
Wear level FTL [38]	SSD	Power loss recovery
ExTENDS [43]	SSD	Performance of throughput for SSD
HIL [44]	SSD	Reliability analysis of the proposed scheme
O1FTL [45]	SSD	Crash recovery time
LeaFTL [51]	SSD	Memory usage of the mapping table, and storage performance
FTL leveraging address mapping [52]	SSD	Response time and analysis of wear balancing percentage
ZoneLife [54]	SSD	SSD longevity, write speed.

The majority of flash storage systems have a FTL (flash translation layer), which is necessary to protect updated metadata from an unexpected power outage. Making the crash recovery scheme efficient is desired since it has a major impact on a flash storage device's writing performance during standard function, as well as its dependability and recovery performance. Even though the rational use of a crash recovery method in a FTL is needed, there are not many publications that address the crash recovery problem in FTL in depth. The authors of [39] introduced a brand-new crash recovery method for a hybrid mapping FTL known as FAST, termed Fast Check (FAST). Increasingly, Fast Check utilizes the features of a FAST FTL, where the log blocks in a log area are employed in a round-robin fashion, to effectively protect the freshly generated address mapping data utilizing periodic checkpoints. As a result, it offers two key benefits over the remaining FTL recovery structures: the first is having a small logging overhead throughout normal FTL operations; and the second is having a quick recovery time in a situation where the log provisioning rate is relatively high, for example, over 20%, and the flash memory capacity is very large, for example, 32 GB or 64 GB.

The authors of [40] propose PORCE, an effective power-off recovery mechanism for flash memory (i.e., a power-off recovery strategy for flash memory). The PORCE minimizes performance deterioration during routine operations since it is closely tied to FTL operations and stores recovery information as sparingly as is feasible. The wear-leveling module is one of the cost-based reclamation methods offered by PORCE. Empirical research demonstrates that PORCE is a successful recovery technique. The recommended method called for distributing the table's store mapping on a spare area of the page in order to be ready for an unexpected loss of power. Instead of creating past PLR innovation, this technique stores the FTL map in a table. Then, flash write codes are included. When there has been a power outage, the same procedure is applied under unusual circumstances. Planning storage solutions as soon as feasible is typically more profitable.

In this [41] FTL, we make extensive use of SW (sequence write) log blocks to improve switch merge performance, while reducing partial merge performance. Additionally, DS-FTL dynamically manages the data blocks and log blocks to lessen the expensive complete merging and wipe operations. Further, in order to maximize the data block's space availabil-

ity, our technique forbids the merging of the data block with a large number of free pages. Our comprehensive experimental findings demonstrate that our suggested strategy (DS-FTL) decreases the number of erase operations while increasing switch merging operations. The DSFTL lowers the overhead associated with trash collection as a consequence.

With its performance and compactness advantages, flash memory has become more common. This makes it possible for it to be quickly adopted as a storage medium in a variety of portable devices, including, but not limited to, smartwatches, smartphones, drones, and in-car entertainment systems. Existing flash translation layers (FTLs) use a cache mapping table (CMT), which only stores a tiny percentage of logical page number to physical page number (LPN-PPN) mappings, to accommodate huge flash storage in such portable devices. During system recovery, for robustness it is crucial to take the CMT reconstruction processes into account. At this time, there is no way to avoid the performance hit caused by unexpected power outages. This is because the delay brought on by discrepancies between the mapping pages in flash storage and the cached RAM page-mapping entries has been disregarded. The best pages to choose for recreating the CMT after device reboots also need to be reviewed. For flash memory storage systems, we address these issues in this work and provide a fault-tolerant power failure recovery mechanism (FTRM). Our empirical investigation demonstrates that FTRM is a reliable recovery technique [42].

To replace the sluggish conventional flash memory solid-state drives, nonvolatile memory phase change memory (NVM-PCM) might be used to create a PCM-SSD (PSSD) (FSSD). To manage PCM endurance limitations, the ability to do in-place updates, bit-addressability, and the ability for it to appear to the host as a block device as its predecessors (HDD and FSSD) do; however, careful design is required, especially for the controller. In [43], the authors suggest implementing ExTENDS, a hardware assumption of NVM-PCM as our future secondary/persistent memory in storage systems, rather than NVM-flash memory. Additionally, the authors offer a PCM file translation layer (Phase FTL) that can effectively handle address translations from a host file system to PCM, while concealing PCM constraints and enabling the PCM blocks to degrade uniformly. Additionally, PCM's bit-addressability and in-place update functionality may be effectively modified by Phase FTL. Our testing reveals that compared to conventional FSSDs, the suggested PCM-SSDs (PSSD) can increase total SSD performance throughput by an average of 69%.

5.3. Mapping Mechanisims i.e., Sector, Block and Hybrid Mapping Schemes in FTL Crash Recovery

Correspondingly, Choi [44] suggested a general FTL form with the hierarchically interacting protocols crash recovery (HIL) module. The HIL framework's primary objective is to treat FTLs' Achilles heel, which is their crash recovery (hence, its name). However, because of its compositional structure, the framework is sufficiently generic to include both block-mapped and page-mapped FTLs, as well as many of their variations, including hybrid variations. Within the HIL paradigm, crash recovery is divided into two stages: structural recovery and functional recovery. Shadow paging is used during structural recovery to atomically remove any remaining effects from program processes running at the time of the crash. Operations that would have been carried out if there had not been a crash are repeated during the functional recovery in a redo-only manner. Due to the idempotent operation of both phases, repeated crashes during recovery will not result in any new issues. By developing a prototype, and demonstrating that its performance during normal execution and also during crash recovery is at least as excellent as that of cutting-edge SSDs, we prove the viability of the proposed HIL architecture.

Due to end-user demands, the capacity of flash-based storage, such as the SSD and embedded multi media card (eMMC), is constantly growing. The flash translation layer (FTL), on the other hand, is in charge of the crash recovery based on the full flash memory if flash-based storage crashes during a power outage. As the capacity of the flash-based storage rises, the recovery time lengthens. We suggest O1FTL, which has an independent

O (1) crash recovery time from flash capacity. The working area approach recommended for the flash file system is adopted by O1FTL, which then tests the design on actual hardware. In terms of I/O performance, the findings demonstrate that O1FTL achieves a crash recovery time that is independent of the capacity and the overhead, and achieves a low P/E cycle [45].

In recent years, flash memory technology has expanded alongside the wave of end-user electronics and fixed systems, with flash memory storage devices' capacity growing quickly. Vendors encounter significant difficulties in system design and analysis in order to effectively control product cost and startup performance. As a result, a crucial area for further research is the timing analysis of system loading, together with crash recovery for a segment-based FTL. Besides, system initialization, crash recovery, and time analysis are the main topics of the current study. The main memory's size and the system startup time are related to the timing study of system initialization. The worst-case recovery time is explained by the temporal analysis of crash recovery. The tests in this work demonstrate that the segment-dependent FTL [46] may be subjected to the timing analysis of system setup and crash recovery.

The authors in [47] model fast metadata recovery and logging approach to support the fast and efficient recovery of power loss. The planned recovery method is based on the page-level mapping FTL. In the constructed FTL, all the metadata modifications are logged into a distinct flash block, which reduces the occurrence of metadata updates. During the recovery phase, only metadata blocks are recovered. The created FTL metadata logging and recovery system support rapid power loss recovery, with a small number of metadata overhead intended for logging.

A new technique that improves the reliability of a page mapping level FTL operating a legacy logging strategy is illustrated in [48]. Moreover, the recovery rate of page mappings also reduces. The novel procedure is to implement a virtually shrunk section and switch off page mapping logs by means of building and storing the sections. This segment proposal also offers embedded NAND flash-based storage to boost its response time. Moreover, the results are also improved, and this innovative plan increases the page mapping reliability, consequently improving the cost of recovery as compared with the conventional page mapping FTL.

In [49], the authors suggest a brand-new FTL system called Lazy FTL, which has great scalability, low response latency, and completely takes away the overhead of merging operations. According to experimental findings, Lazy FTL works better than other commonly used FTL techniques and is extremely near to the theoretically ideal solution. Additionally, we offer a fundamental architecture that helps Lazy FTL recover from system faults.

An efficient plan for the recovery of flash memory using the shadow paging idea was offered in [50] for storage devices utilizing flash memory. The recommended RSLSP solution stores and preserves the map block data as a combination of two tables, i.e., the first block is the original block and the second block is a clone of the original, to battle the problem of unexpected power-off. In comparison to the cutting-edge techniques mentioned in the literature, the proposed strategy not only increases a flash memory device's capacity but also works with the currently used FTL-based methods.

The authors of [51] introduce the Lea FTL, a learning-based flash translation layer (FTL), which uses run-time linear regression to train its address mapping to withstand dynamic data access patterns. The address mapping table's memory footprint is greatly reduced by combining a large number of mapping entries into a learning segment, which also helps SSD controllers' data caching. The Lea FTL also uses a number of optimization strategies, such as the dynamic compaction of learned index segments and out of band metadata verification to accept inaccurate predictions. We put the Lea FTL into practice using an SSD emulator and test it with different storage demands. In comparison to cutting-edge FTL techniques, the Lea FTL reduces the memory usage of the mapping table by an average of 2.9x and boosts storage performance by an average of 1.4x.

In [52], the authors improve the conventional FTL technique and suggest a new FTL approach that, given the upper layer is provided with the address mapping, ensures real-time accomplishment in performance by reducing the worst time of response for the request, regarded as a gauge of the promised flash storage quality of service (QoS). The technique uses a concept i.e., space for time, and distributes the written replica of flash to every write request in order to considerably lower the most unpleasant response time for the request of a write function. The algorithm also takes into account balance factors. Following experimental validation, the proposed approach outperforms active FTL algorithms regarding real-time achievement in performance and improves both the average time of reaction as well as wear balance through more than 11%. The wear-level module is, incidentally, added to an algorithm. The algorithm has been improved, and the worst response time has decreased by 47.6%.

Using the spare area in FTL separated into error code correction (ECC), map information, and reserved, the authors offered a strategy based on page-leveling mapping in this study [53]. The authors compare the recovery time and mapping information management costs of the approach with those of earlier schemes auch as in-block backup and in-page backup, in order to assess the performance of our technology.

In [54], the authors provide an innovative codesign termed Zone Life that exposes the data lifespan semantics from apps to the SSD, in response to the affirmative. The SSD may choose the best error correction code (ECC) from a variety of codes with varying strengths thanks to Zone Life. As a result, the SSD requires a lot less power to store temporary data. Zone Life uses a multigranularity flash translation layer to effectively convert the data addresses of various lifetimes (FTL). Because Zone Life's device firmware makes use of the well-known Zone Namespace interface, and its host driver API generalizes Linux's write hint interface, existing systems may readily adopt it with localized tweaks. Multiple sample databases and cloud workloads are used to test Zone Life, and the findings reveal observable increases in SSD longevity and write speed. Table 4 provides an analysis of various approaches that utilize FTL for crash recovery and leverage its usage. In addition, it provides a taxonomy of these approaches regarding their mapping strategies.

Table 4. Classification of FTL schemes with respect to the mapping mechanism.

Classification Scheme	FTL-Based Approach Analysis with Respect to Their Mapping Scheme
Hybrid mapping [44]	This scheme enables the building of either block mapped as well as page mapped FTLs but also their different hybrids. It manages only a small number of structure blocks.
Hybrid mapping [45]	This scheme was designed based on hybrid mapping. Each chip manages the log area and data area. During operation, both areas are used at the same time.
Block mapping [46]	A crash recovery technique is suggested in this article to prevent incorrect system startup. After a crash, some dirty page map tables might not be written to their appropriate segments, or may be lost altogether.
Hybrid mapping [47]	The FTL metadata consists of root block, which entails block numbers for block map block or metablocks, MapLog block and PageMap block. The 0 and 1 blocks are utilized for root block. First, block 0 is allocated to a root block. However, in the final updated page within the root block, the root information is stored along with locations for metablock data.
Block mapping [48]	The virtually-shrunk sector allows the page level FTL to develop its page mapping organization. Gathering the less significant different sectors can undermine the additional page mapping logs formerly written through the page mapping log formation. Subsequently, the suggested strategy towards page level FTL enhances the cost for page mapping recovery.
Hybrid [45]	Lazy FTL splits the total flash memory to four parts: a data block area (DBA); a cold block area (CBA); a mapping block area (MBA); and an update block area (UBA). The whole parts, apart from the MBA, are employed for storing the data.
Sector mapping [50]	The fundamental concept is to leverage shadow technological benefits and explain them through FTL. The goals are to incorporate shadow algorithms to the FTL algorithm for storage to recover mapping information.
Hybrid mapping [52]	The experimental validation demonstrates that the proposed approach outperforms previous FTL algorithms, with respect to real-time performance, and improves on both wear balance and average response time by more than 11%.

5.4. Significant Considerations in the Design of Power-Off Recovery with FTL

Herein, we describe the important considerations to design the power-off recovery leveraging FTL.

- Designing a power-off recovery system with a flash translation layer (FTL) involves several important factors to ensure the system is reliable and resilient. Herein, we list some key considerations:
- Data consistency: the FTL must ensure that the mapping between physical and logical blocks is consistent, even in the event of a power failure. This requires the use of write buffering, metadata management, and proper error handling to prevent data corruption or loss.
- Write performance: the FTL must minimize the write amplification and ensure that writes are efficient and fast, even in the presence of power failures. This may involve using techniques like wear-leveling and garbage collection to manage flash blocks, and to ensure that writes are consistently distributed throughout the flash memory.
- Error handling: the FTL must be able to handle errors that may occur during power-off, such as bad blocks or block erasures. The FTL must also be able to recover from these errors, for example, by remapping logical blocks to a different physical block.
- Power loss protection: the FTL must use power loss protection (PLP) mechanisms such as supercapacitors or battery-backed SRAM to ensure that data in the write buffer can be flushed to the flash memory, even in the event of a power failure.
- System compatibility: the FTL must be compatible with the inherent flash memory, together with the file system and operating system being used. This may require implementing standard flash translation layer interfaces, such as the NAND flash translation layer (NFTL) or the small computer system interface (SCSI) block commands.

By considering these factors, it is possible to design a power-off recovery system with a flash translation layer that is robust and reliable, and that provides fast and consistent data access, even in the presence of power failures.

6. Conclusions

The flash memory is used in more and more aspects of daily life, and ways to increase its capacity are extensively used in the devices that support it. In the event of an unexpected power loss or power failure, it is crucial to restore data when something goes wrong with flash storage; therefore, crash recovery is an extremely difficult task. It is a vibrant research area that draws many researchers; hence, numerous technical solutions have been suggested thus far. As the performance of flash memory devices is considerably affected by the FTL, it is particularly important to take into consideration the increasing capacity for flash memory devices, resulting in the effectiveness of FTL becoming even more vital. Hence, in this paper, we discuss various optimization schemes for the traditional FTL algorithm, with respect to their flash types and the parameters for measuring performance. These parameters include: the recovery time for a crash; power loss recovery; computational cost; accuracy; reliability evaluation; throughput analysis; the overhead of logging; a decrease of the read latency; write speed; the reliability of page mapping; metadata overhead; the cost of the management of mapping information; and response time. We presented background knowledge related to flash storage types and a software recovery module i.e., the FTL. In addition, we have created a taxonomy of FTL algorithms. Then, we analyzed various recovery schemes and revealed the parameters of comparison for evaluation presented by various schemes. Finally, we have enlisted some vital considerations and recommendations to design an effective power-off recovery utilizing the FTL. Moreover, we provide all the acronyms explanation in Table 5.

Table 5. A list of acronyms and notations.

Acronym/Symbol	Explanation
FTL	Flash translation layer
HIL	Hierarchically interacting protocols crash recovery
DCR	Deterministic crash recovery
SSDs	Solid-state drives
HDD	Hard drive
FAST	Fully associative sector translation
PDA	Personal digital assistants
PORCE	Power-off recovery mechanism
ML	Machine learning
ERF	Error recovery flow
RBER	Raw bit error eate
SVM	Support vector machine
DSFTL	Dynamic setting for FTL
FTRM	Fault tolerant recovery mechanism
ECC	Error correction code
LEAFTL	Learning-based flash translation layer
EMMC	Embedded multi media card
QoS	Quality of service
API	Application programming interface
FSSD	Flash memory solid-state drives
PCM	Phase change memory
NVM	Nonvolatile memory
PSSD	PCM-SSD
OOB	Out of band
LPN	Logical page number
CBA	Cold block area
MBA	Mapping block area
DBA	Data block area
UBA	Update block area
P	Program
E	Erase
SW	Sequential write
CMT	Cache mapping table
DRAM	Dynamic random-access memory
MEW	Metadata embedded write

Author Contributions: Conceptualization, T.S.C. and A.A.; data curation, T.S.C. and A.A.; formal analysis, A.A. and T.S.C.; funding acquisition, T.S.C.; investigation, A.A.; methodology, A.A.; project administration, T.S.C.; resources, T.S.C.; software, A.A.; supervision, T.S.C.; validation, A.A.; writing—original draft, A.A.; writing—review and editing, A.A. and T.S.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2021-0-02051) supervised

by the IITP (Institute for Information & Communications Technology Planning & Evaluation) and the BK21 FOUR program of the National Research Foundation of Korea funded by the Ministry of Education (NRF5199991014091).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tappertzhofen, S. Introduction to non-volatile memory. In *Metal Oxides for Non-Volatile Memory*; Elsevier: Amsterdam, The Netherlands, 2022; pp. 1–32.
2. Suh, S.; Hong, S.M.; Kim, Y.-J.; Park, J.S. A Novel Low Power Method of Combining Saliency and Segmentation for Mobile Displays. *Electronics* **2021**, *10*, 1200. [CrossRef]
3. Bez, R.; Camerlenghi, E.; Modelli, A.; Visconti, A. Introduction to flash memory. *Proc. IEEE* **2003**, *91*, 489–502. [CrossRef]
4. Lu, C.-Y.; Hsieh, K.-Y.; Liu, R. Future challenges of flash memory technologies. *Microelectron. Eng.* **2009**, *86*, 283–286. [CrossRef]
5. Lim, H.J.; Shin, D.; Han, T.H. Parallelism-Aware Channel Partition for Read/Write Interference Mitigation in Solid-State Drives. *Electronics* **2022**, *11*, 4048. [CrossRef]
6. Yang, J.; Lee, S.; Ahn, S. Selective Power-Loss-Protection Method for Write Buffer in ZNS SSDs. *Electronics* **2022**, *11*, 1086. [CrossRef]
7. Kim, S.; Park, H.; Choi, J. Direct-Virtio: A New Direct Virtualized I/O Framework for NVMe SSDs. *Electronics* **2021**, *10*, 2058. [CrossRef]
8. Kinam, K. *Symposium on VLSI-TSA Technology*; IEEE: Piscataway, NJ, USA, 2008; pp. 5–9.
9. Gray, J.; Fitzgerald, B. Flash Disk Opportunity for Server-Applications. Available online: http://www.research.microsoft.com/~gray (accessed on 10 January 2020).
10. Electronics, S. *Nand Flash Memory & Smartmedia Data Book*; Samsung Electronics: Suwon, Republic of Korea, 2007.
11. Amir, B. Flash File System Optimized for Page-Mode Flash Technologies. U.S. Patent No. 5,937,425, 10 August 1999.
12. Chung, T.-S.; Park, D.-J.; Park, S.; Lee, D.-H.; Lee, S.-W.; Song, H.-J. A survey of flash translation layer. *J. Syst. Arch.* **2009**, *55*, 332–343. [CrossRef]
13. Technical Note NAND Flash 101:An Introduction to NAND Flash and How to Design It in to Your Next Product. Available online: https://user.eng.umd.edu/~blj/CS-590.26/micron-tn2919.pdf (accessed on 1 January 2023).
14. Gupta, A.; Kim, Y.; Urgaonkar, B. DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings. In Proceedings of the 14th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'09), Washington, DC, USA, 7–11 March 2009; pp. 229–240.
15. Kang, J.-U.; Jo, H.; Kim, J.-S.; Lee, J. A Superblock-Based Flash Translation Layer for NAND Flash Memory. In Proceedings of the 6th ACM & IEEE International Conference on Embedded Software—EMSOFT '06, Seoul, Republic of Korea, 22–25 October 2006; IEEE Press: New York, NY, USA, 2006; pp. 161–170.
16. Kim, J.; Kim, J.M.; Noh, S.; Min, S.L.; Cho, Y. A space-efficient flash translation layer for CompactFlash systems. *IEEE Trans. Consum. Electron.* **2002**, *48*, 366–375.
17. Lee, S.; Shin, D.; Kim, Y.-J.; Kim, J. LAST: Locality-Aware Sector Translation for NAND Flash Memory-Based Storage Systems. *SIGOPS Oper. Syst. Rev.* **2008**, *42*, 36–42. [CrossRef]
18. Lee, S.-W.; Park, D.-J.; Chung, T.-S.; Lee, D.-H.; Park, S.; Song, H.-J. A log buffer-based flash translation layer using fully-associative sector translation. *ACM Trans. Embed. Comput. Syst.* **2007**, *6*, 18. [CrossRef]
19. Lim, S.-P.; Lee, S.-W.; Moon, B. FASTer FTL for Enterprise-Class Flash Memory SSDs. In Proceedings of the IEEE SNAPI 2010 6th International Workshop on Storage Network Architecture and Parallel I/Os, New York, NY, USA, 3 May 2010; pp. 3–12.
20. Mativenga, R.; Paik, J.-Y.; Kim, Y.; Lee, J.; Chung, T.-S. RFTL: Improving performance of selective caching-based page-level FTL through replication. *Clust. Comput.* **2018**, *22*, 25–41. [CrossRef]
21. Wu, C.-H. A self-adjusting flash translation layer for resource-limited embedded systems. *ACM Trans. Embed. Comput. Syst.* **2010**, *9*, 1–26. [CrossRef]
22. Chen, N.; Chen, B.; Shi, W. A Cross-layer Plausibly Deniable Encryption System for Mobile Devices. In Proceedings of the 18th EAI International Conference on Security and Privacy in Communication Networks (SecureComm'22), Kansas City, MI, USA, 17–19 October 2022.
23. Pan, Y.; Li, Y.; Zhang, H.; Chen, H.; Lin, M. GFTL: Group-level mapping in flash translation layer to provide efficient address translation for NAND flash-based SSDs. *IEEE Trans. Consum. Electron.* **2020**, *66*, 242–250. [CrossRef]

24. Ramasamy, A.S.; Karantharaj, P. Persistent pre-allocation and back link based write back journaling technique. In Proceedings of the 2013 IEEE International Conference of IEEE Region 10 (TENCON 2013), Xi'an, China, 22–25 October 2013; IEEE: Piscataway, NJ, USA; pp. 1–6.
25. Chung, T.-S.; Park, D.-J.; Park, S.; Lee, D.-H.; Lee, S.-W.; Song, H.-J. System software for flash memory: A survey. In Proceedings of the International Conference on Embedded and Ubiquitous Computing, Seoul, Republic of Korea, 1–4 August 2006.
26. Kumar, S.; Dubey, K.; Singh, P.K. A survey on flash translation layer for NAND flash memory. *Indian J. Sci. Technol.* **2018**, *11*, 1–7. [CrossRef]
27. Luo, Y.; Lin, M. Flash translation layer: A review and bibliometric analysis. *Int. J. Intell. Comput. Cybern.* **2021**, *14*, 480–508. [CrossRef]
28. Luo, Y.; Lin, M.; Pan, Y.; Xu, Z. Dual locality-based flash translation layer for NAND flash-based consumer electronics. *IEEE Trans. Consum. Electron.* **2022**, *68*, 281–290. [CrossRef]
29. Kwon, S.J.; Ranjitkar, A.; Ko, Y.B.; Chung, T.S. FTL algorithms for NAND-type flash memories. *Des. Autom. Embed. Syst.* **2011**, *15*, 191–224. [CrossRef]
30. Lee, S.; Jee, J.; Park, H. Machine Learning-Based Error Recovery System for NAND Flash Memory with Process Variation. In Proceedings of the 2021 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 20–22 October 2021; pp. 1537–1541. [CrossRef]
31. Kang, B.; Jee, J.; Park, H. Intelligent error recovery flow prediction for low latency NAND flash memory system. In Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Republic of Korea, 21–23 October 2020; pp. 1367–1372.
32. Santikellur, P.; Buddhanoy, M.; Sakib, S.; Ray, B.; Chakraborty, R.S. A shared page-aware machine learning assisted method for predicting and improving multi-level cell NAND flash memory life expectancy. *Microelectron. Reliab.* **2023**, *140*, 114867. [CrossRef]
33. Hwang, M.; Jee, J.; Kang, J.; Park, H.; Lee, S.; Kim, J. Dynamic Error Recovery Flow Prediction Based on Reusable Machine Learning for Low Latency NAND Flash Memory Under Process Variation. *IEEE Access* **2022**, *10*, 117715–117731. [CrossRef]
34. Jung, S.; Song, Y.H. Data loss recovery for power failure in flash memory storage systems. *J. Syst. Archit.* **2015**, *61*, 12–27. [CrossRef]
35. Nazari, M.; Zadeh, R.T.; Asghari, S.A.; Marvasti, M.B.; Rahmani, A.M. FRCD: Fast recovery of compressible data in flash memories. *Comput. Electr. Eng.* **2019**, *78*, 520–535. [CrossRef]
36. Chen, R.; Zhang, C.; Wang, Y.; Shen, Z.; Liu, D.; Shao, Z.; Guan, Y. DCR: Deterministic crash recovery for NAND flash storage systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *38*, 2201–2214. [CrossRef]
37. Moon, S.; Lim, S.P.; Park, D.J.; Lee, S.W. Crash recovery in FAST FTL. In *IFIP International Workshop on Software Technolgies for Embedded and Ubiquitous Systems*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 13–22.
38. Tran, V.D.; Park, D.-J. A survey of data recovery on flash memory. *Int. J. Electr. Comput. Eng.* **2020**, *10*, 2088–8708. [CrossRef]
39. Park, J.-H.; Park, D.-J.; Chung, T.-S.; Lee, S.-W. A Crash Recovery Scheme for a Hybrid Mapping FTL in NAND Flash Storage Devices. *Electronics* **2021**, *10*, 327. [CrossRef]
40. Chung, T.S.; Lee, M.; Ryu, Y.; Lee, K. PORCE: An efficient power off recovery scheme for flash memory. *J. Syst. Archit.* **2008**, *54*, 935–943. [CrossRef]
41. Chae, S.-J.; Mativenga, R.; Paik, J.-Y.; Attique, M.; Chung, T.-S. DSFTL: An Efficient FTL for Flash Memory Based Storage Systems. *Electronics* **2020**, *9*, 145. [CrossRef]
42. Mativenga, R.; Hamandawana, P.; Chung, T.-S.; Kim, J. FTRM: A Cache-Based Fault Tolerant Recovery Mechanism for Multi-Channel Flash Devices. *Electronics* **2020**, *9*, 1581. [CrossRef]
43. Mativenga, R.; Hamandawana, P.; Kwon, S.J.; Chung, T. ExTENDS: Efficient Data Placement and Management for Next Generation PCM-Based Storage Systems. *IEEE Access* **2019**, *7*, 148718–148730. [CrossRef]
44. Choi, J.-Y.; Nam, E.H.; Seong, Y.J.; Yoon, J.; Lee, S.; Kim, H.; Park, J.; Woo, Y.-J.; Lee, S.; Min, S.L. HIL: A Framework for Compositional FTL Development and Provably-Correct Crash Recovery. *ACM Trans. Storage* **2018**, *14*, 1–29. [CrossRef]
45. Park, J.Y.; Park, H.; Yoo, C. Design and Implementation of Flash Translation Layer with O (1) Crash Recovery Time. *KIISE Trans. Comput. Pract.* **2015**, *21*, 639–644. [CrossRef]
46. Wu, C.H.; Lin, H.H. Timing analysis of system initialization and crash recovery for a segment-based flash translation layer. *ACM Trans. Des. Autom. Electron. Syst. (TODAES)* **2012**, *17*, 1–21. [CrossRef]
47. Lim, S.H. Implementation of metadata logging and power loss recovery for page-mapping FTL. *IEICE Electron. Express* **2013**, *10*, 20130339. [CrossRef]
48. Kim, J.H. Rapid Recovery by Maximizing Page-Mapping Logs Deactivation. *IEICE Trans. Inf. Syst.* **2021**, *104*, 885–889. [CrossRef]
49. Ma, D.; Feng, J.; Li, G. LazyFTL: A page-level flash translation layer optimized for NAND flash memory. In Proceedings of the 2011 international conference on Management of data—SIGMOD '11, Athens, Greece, 12–16 June 2011. [CrossRef]
50. Alahmadi, A.; Chung, T.S. RSLSP: An Effective Recovery Scheme for Flash Memory Leveraging Shadow Paging. *Electronics* **2022**, *11*, 4126. [CrossRef]
51. Sun, J.; Li, S.; Sun, Y.; Sun, C.; Vucinic, D.; Huang, J. LeaFTL: A Learning-Based Flash Translation Layer for Solid-State Drives. *arXiv* **2022**, arXiv:2301.00072.
52. He, Q.; Bian, G.; Zhang, W.; Li, Z. RTFTL: Design and implementation of real-time FTL algorithm for flash memory. *J. Supercomput.* **2022**, *78*, 18959–18993. [CrossRef]

53. Dai, T.V.; Park, J.; Park, D.J. A Novel Recovery Data Technique on MLC NAND Flash Memory. In Proceedings of the 2019 11th International Conference on Knowledge and Systems Engineering (KSE), Da Nang, Vietnam, 24–26 October 2019; pp. 1–5. [CrossRef]
54. Chen, Y.C.; Wu, C.F.; Chang, Y.H.; Kuo, T.W. ZoneLife: How to Utilize Data Lifetime Semantics to Make SSDs Smarter. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**. [CrossRef]