Evaluating File System Reliability on Solid State Drives

班级：国光硕 1905　姓名：周瀚　学号：M201973016　论文号：05

1 背景

近年来，由于良好的性能表现，固态硬盘(Solid State Drive)在市场上日益流行，逐渐代替传统机械硬盘(Hard Disk Drive)的主流地位。随着固态硬盘的广泛采用，存储系统的可靠性依赖于这些新设备的可靠性，以及运行在设备上的文件系统处理可能产生的错误的能力。

此前由 Vijayan Prabhakaran 等人于 2005 年发布的经典论文[45]详细地研究了机械硬盘上的三个文件系统的可靠性。然而，机械硬盘与固态硬盘的故障模式存在较大差别，例如最近的研究[39, 43, 48]表明，固态硬盘读写时发生局部故障的概率比机械硬盘高一个数量级，而替换概率则低一个数量级；其他研究表明，相比机械硬盘的固件，固态硬盘的 FTL 固件更容易出现故障，尤其在异常断电时。同时，从 2005 年到现在，文件系统也有了显著的发展，现在的文件系统引入了许多高级特性，如 ext4 的日志机制、Btrfs 的写时拷贝，以及 F2FS 针对 Flash 的数据存储进行了专门优化等。

因此本文的作者认为在 13 年后，有必要基于现在的文件系统的新特性，对文件系统和固态硬盘的可靠性特征进行重新研究。本文描述了 ext4、Btrfs 和 F2FS 三个文件系统对固态硬盘故障检测与恢复能力的测试过程，并根据测试结果，对固态硬盘文件系统提出了一些设计思想和建议。

2 测试方法

通过模拟不同类型的 SSD 故障，判断文件系统哪一部分受到影响，从而分析文件系统检测与恢复故障的能力。测试只限定于运行在单个驱动器上的本地文件系统，不考虑 RAID 机制；只考虑局部驱动器故障，即部分操作受影响，而 SSD 没有永久失效。

2.1 故障分类

下表中的行对应不同的 Flash 级错误类型，列对应文件系统级故障类型，‘√’单元格表示该行的 Flash 级错误可能导致对应列的文件系统级故障。

SSD/Flash Errors	(a)	(b)	(c)	(d)	(e)
Uncorrectable Bit Corruption	✓				
Silent Bit Corruption			✓		
FTL Metadata Corruption	✓	✓	✓		
Misdirected Writes	✓		✓		
Shorn Writes				✓	
Dropped Write	✓	✓	✓		✓
Incomplete Program Operation			✓	✓	
Incomplete Erase Operation			✓		

Table 1 Flash Error **在文件系统上的表现：**

(a) Read I/O Error; (b) Write I/O Error; (c) Corruption; (d) Shorn Write; (e) Lost Write.

Uncorrectable Bit Corruption：Flash 的数据保持错误、读干扰、写干扰、Block 损耗或故障等原因导致的超出 ECC 纠错能力的位损坏。最近的研究显示，ECC 不可纠正的位损坏发生概率高于可被纠正的未损坏[48]。当应用程序试图访问受影响的数据时，驱动器将返回读 I/O 错误。

Silent Bit Corruption：驱动器在未检测出错误的情况下，将损坏的数据返回给应用程序。在[53]的研究中，15 个驱动模型中有 3 个在电源故障的情况下经历了静默数据损坏。

FTL Metadata Corruption：FTL 元数据出现静默数据损坏，则应用程序访问时，驱动器可能对一个不正确或不存在的块进行读写。

Misdirected Writes：SSD 内部执行写操作时，数据被写入一个不正确的位置。

Shorn Writes：文件系统发出的写操作仅完成了一部分。可能的原因包括逻辑块与物理块大小的不匹配，或异常断电等。

Dropped Write：显式刷新缓存后，SSD 内部的写操作被丢弃，例如异常断电时 SSD 缓存中的数据尚未持久化。当与 FTL 元数据相关时，则随后的数据访问可能表现出读写 I/O 错误或损坏；当与文件数据相关时，表现为文件系统从未发起写操作。

Incomplete Program Operation：Flash 编程操作未完成，FTL 未检测到。

Incomplete Erase Operation：Flash 块的擦除操作未完成，FTL 未检测到。

2.2 错误注入

利用 Linux(内核版本 4.17)的 device mapper 框架，创建一个虚拟块设备，拦截文件系统与物理设备之间请求并跟踪，还可通过对请求内容的修改，来模拟不同类型的 SSD 故障，观察文件系统的反应。对五类故障的模拟方式如下：

Read I/O Error 和 Write I/O Error：模块拦截读/写请求，获取请求类型、块号和数据结构类型等信息，向文件系统返回错误代码；

Corruption：破坏块内特定的数据结构、字段和字节；

Shorn Write：[53]的作者观察到，部分写仅会以块的前 3/8 或 7/8 两种情况写入，因此这里只保留写入块(4KB)的前 3/8，其余部分为 0；

Lost Write：删除一个或多个块；

该错误注入模块可扩展新的故障模式，还可扩展其包含的文件系统种类。

2.3 测试步骤

(1) 初始化文件系统，填充数据；
(2) 运行一个测试程序，利用 blktrace 和 device mapper 来捕获 trace，获得实际访问的块；
(3) 使用 dumpe2fs、btrfs-inspect、dump.F2FS 等工具检查硬盘内容变化，识别块类型和块内数据结构；
(4) 重新初始化硬盘镜像；
(5) 使用相同的模块、工具、程序，将一种错误注入块或数据结构中后再次测试，观察文件系统对故障的反应；
(6) 测试程序结束后卸载文件系统，使用文件系统的完整性检查器 fsck，比较两次测试后硬盘镜像的变化。

其中，测试程序来自于以下 Linux 系统调用: mount, umount, open, creat, access, stat, lstat, chmod, chown, utime, rename, read, write, truncate, readlink, symlink, unlink, chdir, rmdir, mkdir, getdirentries, chroot.

文件系统对故障的反应与 fsck 检查结果的分类如下表。

Symbol	Level	Description
○	D_{Zero}	No detection.
–	$D_{ErrorCode}$	Check the error code returned from the lower levels.
\	D_{Sanity}	Check for invalid values within the contents of a block.
/	$D_{Redundancy}$	Checksums, replicas, or any other form of redundancy.
|	D_{Fsck}	Detect error using the system checker.
○	R_{Zero}	No attempt to recover.
/	R_{Retry}	Retry the operation first before returning an error.
|	$R_{Propagate}$	Error code propagated to the user space.
\	$R_{Previous}$	File system resumes operation from the state exactly before the operation occurred.
–	R_{Stop}	The operation is terminated (either gracefully or abruptly); the file system may be mounted as read-only.
■	R_{Fsck_Fail}	Recovery failed, the file system cannot be mounted.
■	$R_{Fsck\_Partial}$	The file system is mountable, but it has experienced data loss in addition to operation failure.
■	R_{Fsck_Orig}	Current operation fails, file system restored to pre-operation state.
■	R_{Fsck_Full}	The file system is fully repaired and its state is the same with the one generated by the execution where the operation succeeded without any errors.

Table 2 **文件系统检测与恢复分类**

例如，发生 I/O 错误时，存储在块中的数据不会传递到硬盘或文件系统，因此文件系统不会进行完整性检查，不返回 D_{Sanity}；对于静默错误，由于文件系统无法检测出，故不返回 $D_{ErrorCode}$。

3 测试结果

测试结果被组织为六类，其中 Shorn Write 的检测与恢复分为两个场景，再次访问被持久化的部分数据的程序分别是应用程序和 fsck。下表表示执行 a~w 列的程序时，遇到相应的故障，行对应的数据结构受到影响，在此情况下文件系统做出的反应，以及 fsck 的检查结果，其符号表示如 Table 2 所示。

其中，a~w 列对应的程序分别为：(a) access, (b) truncate, (c) open, (d) chmod, (e) chown, (f) utimes, (g) read, (h) rename, (i) stat, (j) lstat, (k) readlink, (l) symlink, (m) unlink, (n) chdir, (o) rmdir, (p) mkdir, (q) write, (r) getdirentries, (s) creat, (t) mount, (v) umount, (w) chroot.

3.1 Btrfs

Btrfs 是一个写时拷贝的文件系统，所有元数据都使用 B-Tree 存储，并使用软 RAID 与校验和进行保护，还拥有快照、克隆、透明压缩等特性。

从下图可以看出，Btrfs 始终能够检测出所有 I/O 错误和 Corruption 故障，通过大量的校验和实现。然而 Btrfs 从故障中恢复的成功率较低，六种故障模式中有四种可能导致内核崩溃，甚至运行 fsck 后也无法挂载文件系统。

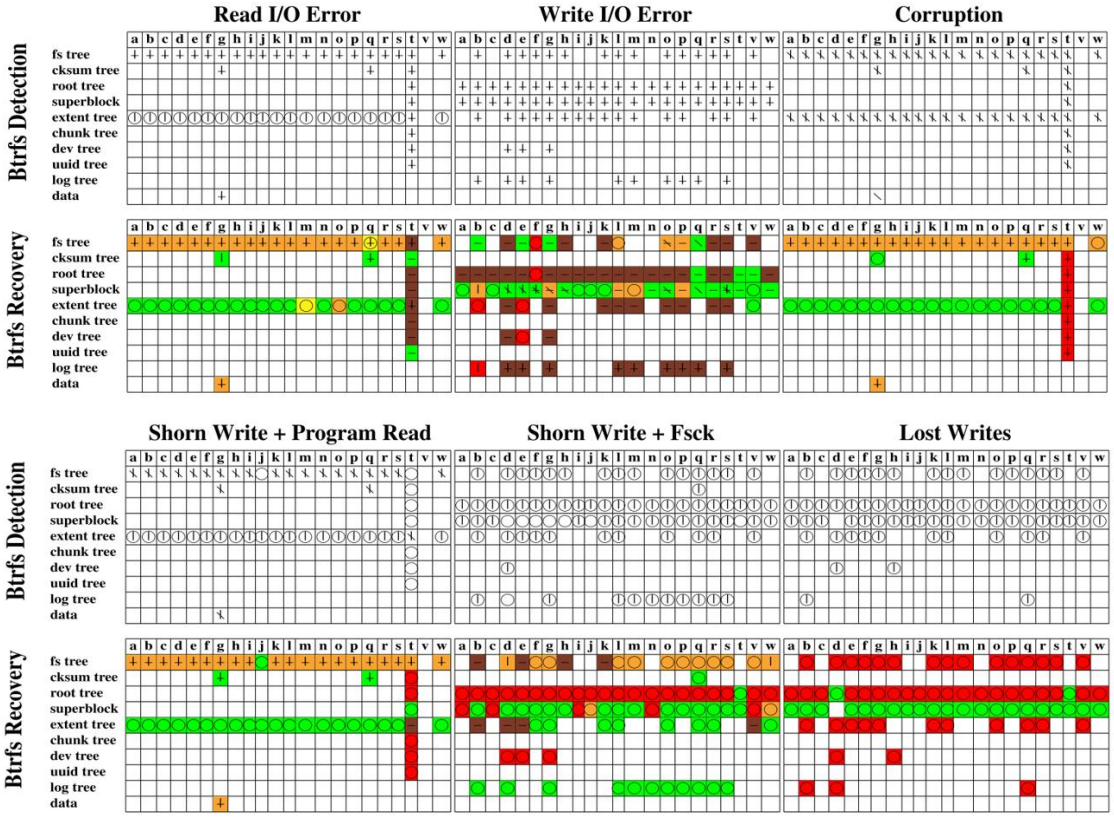

Figure 1 The results of our analysis on the detection and recovery policies of Btrfs

3.1.1 Read I/O Error

所有错误都能够被检测出，操作中止，并在操作系统的消息日志中注册。Fsck 能够在大多数情况下检测并纠正错误，除了两种情况：fstree 结构受到影响，fsck 删除不可读的块，返回错误码；挂载过程中访问关键树时发生错误，挂载失败，fsck 也无法修复。

3.1.2 Corruption

利用树节点的校验和能够可靠地检测错误。Btrfs 根据底层设备类型采用不同的恢复协议，设备为 HDD 时将调用 Btrfs-scrub 读取副本来恢复损坏的块，设备为 SSD 时则默认关闭元数据备份，因为 SSD 可以在内部将主块及其副本重映射到单个物理位置，从而消除重复，且 SSD 控制器短时间内可能将写入的数据放入同一物理存储单元(即单元、擦除块等)，可能该单元已经损坏[7]。这种设计选择会导致即使只有 1 比特数据发生错误，整个树节点也会被删除，在挂载时发生错误将产生更严重的后果，操作失败，驱动器处于不一致状态，即使调用 fsck 也无法恢复。

目录损坏时，虽然 Btrfs 使用两个独立数据结构来存储目录，但是一个受损时却并没有用另一个来恢复目录，这是令人惊讶的。

3.1.3 Write I/O Error

Brrfs 的超级块有多个副本，更新时保持一致。主超级块发生错误时，操作中止，fs 重新挂载为只读；副本发生错误时操作顺利完成，但没有被更新，违反了超级块之间的隐式一致性。

树节点发生错误时，在消息日志中记录，但由于文件系统的异步性，错误码没有直接返回给用户程序，大部分情况下文件系统将被挂载为只读，在修复模式下卸载、运行 fsck 将使 extent tree、log tree、root tree 和 fs tree 的根节点不可读。

3.1.4 Shorn Write + Program Read

与 Corruption 故障类似，唯一的例外是超级块，因为超级块的大小小于物理块的 3/8，因此不会受影响。

3.1.5 Shorn Write + Fsck

root tree 发生 Shorn Write 时，fsck 无法恢复。

3.1.6 Lost Write

文件系统无法检测到故障，fsck 可以，但除了超级块以外，其他数据结构受影响时无法恢复，并导致文件系统无法挂载。

3.2 ext4

ext4 是目前在 Linux 和 Android 系统上使用最为广泛的文件系统。日志机制使得 ext4 能够从大量故障场景中恢复。在读取目录、inode 和 extent 等数据结构时，ext4 部署了丰富的完备性检查，有助于处理 Corruption 故障。另外，ext4 默认不使用校验和。

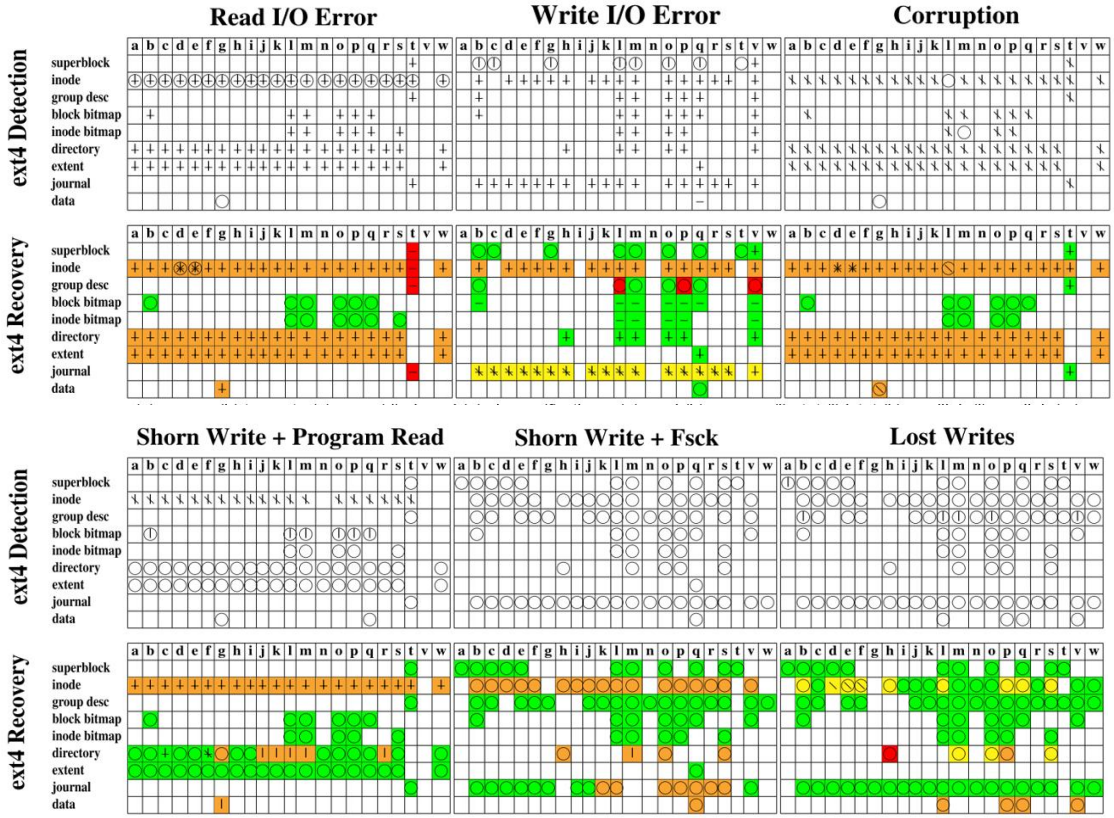

Figure 2 The results of our analysis on the detection and recovery policies of ext4

3.2.1 inode 的 I/O Error, Corruption 和 Shorn Write

最容易造成数据丢失的场景，存储在 inode 对应的块内数据均无法访问，但文件系统仍处于一致性状态。

3.2.2 目录块的 Read I/O Error, Corruption 和 Shorn Write

文件系统能够检测出错误，并最终删除相应的目录和文件。默认情况下，空文件被删除，非空文件被移动到 lost+found 目录下，父子关系丢失。

3.2.3 组描述符的 Write I/O Error

大部分情况下可检测并恢复，除了一种情况：fsck 尝试重新构建组描述符，并将其写入原位，则可能发生相同的错误而陷入循环。

3.2.4 挂载时发生 Read I/O Error

如果在读取关键元数据结构时发生读 I/O 错误，ext4 将无法完成挂载操作。在这种情况下，即使在调用 fsck 也不能挂载文件系统。

3.3 F2FS

F2FS 是专为 Flash 设计的日志结构文件系统。数据和元数据默认被写入 6 个活动日志，按数据/元数据、文件/目录和其他启发式方法分组，允许将相似的块放在一起以提高顺序写比率。文件系统的元数据区域存储在固定位置，包括检查点(CP)、段信息表(SIT)和节点地址表(NAT)等，F2FS 对这三个数据结构使用双位置方法，一个用于在挂载期间初始化文件系统的状态，而另一个是在文件系统运行期间更新的影子副本。

F2FS 几乎在所有情况下都可以检测出 Read I/O Error，并通知用户程序，但始终无法检测到 Write I/O Error，此外也无法有效处理 Shorn Write 和 Lost Write 故障。

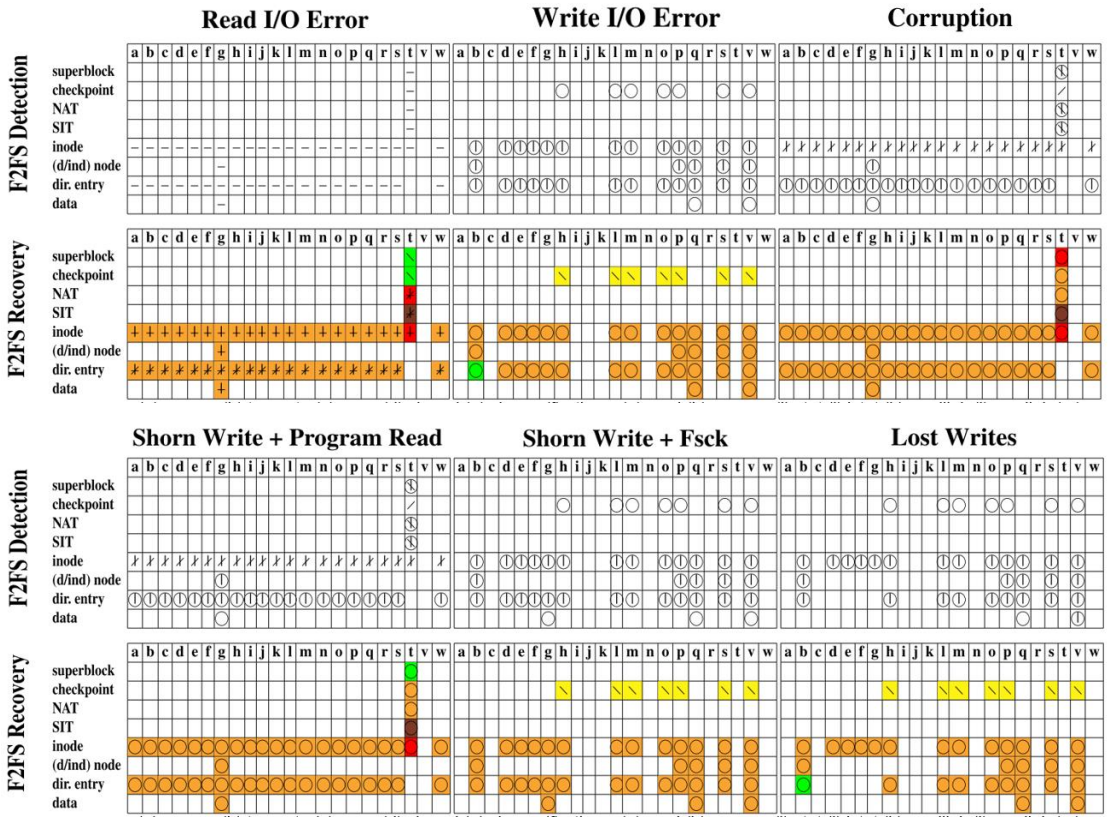

Figure 3 The results of our analysis on the detection and recovery policies of F2FS

3.3.1 Read I/O Error

挂载时，如果读取检查点、NAT(Node Address Table)、SIT(Segment Information Table)发生故障，则文件系统挂载为只读；如果根目录 inode 不能访问，则文件系统无法挂载。通常 fsck 并不能使文件系统从错误中恢复，因为每次无法从磁盘读取一个块时，fsck 都会使用

断言错误终止操作。

3.3.2 Write I/O Error 和 Lost Write

F2FS 没有检测出 Write I/O Error，因此将导致 Corruption、读取到垃圾数据和潜在的数据丢失等问题，产生与 Lost Write 相同的结果。由于 fsck 的恢复协议不完善，最终数据丢失。

3.3.3 Corruption

文件系统仅能够检测到 inode 和检查点的错误，因为它们受到校验和的保护，但 fsck 无法保证它们完全从错误中恢复，依然可能导致文件和其中存储的数据的丢失。

inode 页脚信息损坏，fsck 将直接丢弃；与目录相关的 inode 损坏，目录中的所有文件和子目录都被 fsck 标记为无法找到，其中有效 inode 被移动到 lost_found 目录下，仅为文件创建条目，不包括子目录，不同子目录下的同名文件将相互覆盖。

F2FS 为检查点建立备份，只有当两个副本均出现错误时，文件系统将无法挂载。

其他数据结构的损坏可能导致更严重的后果。

虽然超级块有两个副本，但是文件系统对超级块的损坏检测完全依赖于对其(大多数)字段执行的一组完备性检查，而不是两个副本的校验和或比较。如果完整性检查标识了无效的值，则使用备份副本进行恢复。然而，测试结果表明，完备性检查不能检测所有无效值。因此，根据损坏的字段，文件系统的可靠性可能受到影响，例如用于计算检查点起始地址的偏移字段损坏，会导致文件系统在挂载期间从无效的检查点位置引导，最终在卸载期间挂起。

SIT 与 NAT 均没有设置任何形式的冗余来保护。SIT 的错误将严重损坏文件系统的一致性，最终导致无法挂载。Fsck 会将 NAT 的错误条目标记为不可访问，原条目指向的文件被移动到 lost_found 目录下，与上面提到的相同，同名文件相互覆盖。

目录项的存储被组织成块，目前还没有块的错误检测机制。其中部分字段错误时，文件系统将向用户程序返回垃圾数据；inode 节点 ID 字段错误时，可能指向当前存储的任何节点；若 NAT 位图中相应的索引出错，则原条目可能消失。最终受影响的条目将被 fsck 标记为无法访问，与上面提到的处理机制相同，文件被移动到 lost_found 目录下，同名文件相互覆盖。

3.3.4 Shorn Write + Program Read

除了超级块之外，测试结果同 Corruption 类似，因为 Shorn Write 可以被视为一种特殊的 Corruption。

3.3.5 Shorn Write + Fsck

目录项所在块没有受到保护，出现部分写时无法检测到，可能导致以下几个问题：首先，文件系统的有效条目消失，包括指向当前目录及其父目录的特殊条目；其次，在某些情况下，重新挂载文件系统之后，尝试列出目录的内容时陷入死循环。在这两种情况下，受影响的条目最终都被 fsck 标记为无法访问，文件被转储到 lost_found 目录中，同名文件相互覆盖。

4 总结

从三个文件系统的测试结果中可以看出，ext4 在 Corruption 和 I/O 错误的检测与恢复上，相比 ext3 有显著的改进，在三个文件系统中表现最佳。Btrfs 和 F2FS 的故障检测与恢复机制都存在致命漏洞，Btrfs 是一个有快照和克隆等高级特性的生产级文件系统，具有良好的故障检测机制，但是当关键数据结构(如元数据)错误时无法恢复，部分原因是对 SSD 默认禁用了元数据复制。F2FS 的开发在很长时间内集中于性能提升，对可靠性缺乏关注，在测试中对故障的检测与恢复能力最弱。

文件系统并非总是充分利用现有的冗余机制，例如 Btrfs 对目录建立两个独立数据结构以提升性能，但没有用于错误恢复。元数据出错时更容易导致严重故障，文件系统应尽可能通过完整性检查来保护元数据，在这方面 ext4 比 F2FS 做得更彻底。校验和是双刃剑，虽然有助于提高错误检测，但粒度过大可能导致严重的数据丢失，粒度过小则会造成存储和性能开销。Flash 系统中不能排除这种情况，主块和备份块在同一故障域中(相同的擦除块或相同的闪存芯片)，并且在很短的时间间隔内同时写入时，冗余的效果可能较差。

然而，实验中对于文件系统级故障的分类可能有待商榷，例如除了读写 I/O 错误之外，其他故障类型均是静默的，其中 Corruption 包含范围较为广泛，Shorn Write、Lost Write 均可视为 Corruption 的特殊情况，可能存在概念上的重合。此外，因为超级块通常小于 4KB 的 3/8，所以对于超级块的 Shorn Write 模拟没有生效，实验中并未对超级块受到部分写的影响的场景进行测试。

参考文献

[1] Btrfs Bug Report. https://bugzilla.kernel.org/show_bug.cgi?id=198457.

[2] F2FS Bug Report. https://bugzilla.kernel.org/show_bug.cgi?id=200635.

[3] F2FS Bug Report - Write I/O Errors. https://bugzilla.kernel.org/show_bug.cgi?id=200871.

[4] F2FS Patch File. https://sourceforge.net/p/linux-F2FS/mailman/message/36402198/.

[5] fs-verity: File System-Level Integrity Protection. https://www.spinics.net/lists/linux-fsdevel/msg121182.html. [Online; accessed 06-Jan-2019].

[6] Github code repository. https://github.com/uoftsystems/dm-inject.

[7] Btrfs mkfs man page. https://btrfs.wiki.kernel. org/index.php/Manpage/mkfs.btrfs, 2019. [Online; accessed 06-Jan-2019].

[8] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D Davis, Mark S Manasse, and Rina Panigrahy. Design tradeoffs for SSD performance. In USENIX Annual Technical Conference (ATC '08), volume 57, 2008.

[9] Lakshmi N Bairavasundaram, Andrea C ArpaciDusseau, Remzi H Arpaci-Dusseau, Garth R Goodson, and Bianca Schroeder. An Analysis of Data Corruption in the Storage Stack. ACM Transactions on Storage (TOS), 4(3):8, 2008.

[10] Hanmant P Belgal, Nick Righos, Ivan Kalastirsky, Jeff J Peterson, Robert Shiner, and Neal Mielke. A new reliability model for post-cycling charge retention of flash memories. In

Proceedings of the 40th Annual International Reliability Physics Symposium, pages 7–20. IEEE, 2002.
[11] Matias Bjørling, Javier Gonzalez, and Philippe Bonnet. LightNVM: The Linux Open-Channel SSD Subsystem. In Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST '17), pages 359–374, Santa Clara, CA, 2017. USENIX Association.
[12] Simona Boboila and Peter Desnoyers. Write Endurance in Flash Drives: Measurements and Analysis. In Proceedings of the 8th USENIX Conference on File and Storage Technologies (FAST '10), pages 115–128. USENIX Association, 2010.
[13] Adam Brand, Ken Wu, Sam Pan, and David Chin. Novel read disturb failure mechanism induced by FLASH cycling. In Proceedings of the 31st Annual International Reliability Physics Symposium, pages 127–132. IEEE, 1993.
[14] Yu Cai, Saugata Ghose, Erich F Haratsch, Yixin Luo, and Onur Mutlu. Error characterization, mitigation, and recovery in flash-memory-based solid-state drives. Proceedings of the IEEE, 105(9):1666–1704, 2017.
[15] Yu Cai, Saugata Ghose, Yixin Luo, Ken Mai, Onur Mutlu, and Erich F Haratsch. Vulnerabilities in MLC NAND flash memory programming: experimental analysis, exploits, and mitigation techniques. In 23rd International Symposium on High-Performance Computer Architecture (HPCA), pages 49–60. IEEE, 2017.
[16] Yu Cai, Erich F Haratsch, Onur Mutlu, and Ken Mai. Error patterns in MLC NAND flash memory: Measurement, Characterization, and Analysis. In Proceedings of the Conference on Design, Automation and Test in Europe, pages 521–526. EDA Consortium, 2012.
[17] Yu Cai, Yixin Luo, Erich F Haratsch, Ken Mai, and Onur Mutlu. Data retention in MLC NAND flash memory: Characterization, optimization, and recovery. In 21st International Symposium on High Performance Computer Architecture (HPCA), pages 551–563. IEEE, 2015.
[18] Yu Cai, Onur Mutlu, Erich F Haratsch, and Ken Mai. Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation. In 31st International Conference on Computer Design (ICCD), pages 123–130. IEEE, 2013.
[19] Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F Haratsch, Adrian Cristal, Osman S Unsal, and Ken Mai. Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime. In 30th International Conference on Computer Design (ICCD), pages 94–101. IEEE, 2012.
[20] Jinrui Cao, Om Rameshwar Gatla, Mai Zheng, Dong Dai, Vidya Eswarappa, Yan Mu, and Yong Chen. PFault: A General Framework for Analyzing the Reliability of High-Performance Parallel File Systems. In Proceedings of the 2018 International Conference on Supercomputing, pages 1–11. ACM, 2018.
[21] Paolo Cappelletti, Roberto Bez, Daniele Cantarelli, and Lorenzo Fratin. Failure mechanisms of Flash cell in program/erase cycling. In Proceedings of the IEEE International Electron Devices Meeting, pages 291–294. IEEE, 1994.
[22] Feng Chen, David A. Koufaty, and Xiaodong Zhang. Understanding Intrinsic Characteristics and System Implications of Flash Memory Based Solid State Drives. In Proceedings of the 2009 ACM SIGMETRICS International Conference on Measurement and Modeling of USENIX Association 2019 USENIX Annual Technical Conference 795Computer Systems (SIGMETRICS '09), pages 181–192, 2009.
[23] Robin Degraeve, F Schuler, Ben Kaczer, Martino Lorenzini, Dirk Wellekens, Paul Hendrickx,

Michiel van Duuren, GJM Dormans, Jan Van Houdt, L Haspeslagh, et al. Analytical percolation model for predicting anomalous charge loss in flash memories. IEEE Transactions on Electron Devices, 51(9):1392–1400, 2004.
[24] Jake Edge. File-level Integrity. https://lwn.net/Articles/752614/, 2018. [Online; accessed 06-Jan2019].
[25] Aishwarya Ganesan, Ramnatthan Alagappan, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Redundancy Does Not Imply Fault Tolerance: Analysis of Distributed Storage Reactions to Single Errors and Corruptions. In Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST '17), pages 149–166, Santa Clara, CA, 2017. USENIX Association.
[26] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, and J. K. Wolf. Characterizing Flash Memory: Anomalies, Observations, and Applications. In 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pages 24–33, Dec 2009.
[27] Laura M Grupp, John D Davis, and Steven Swanson. The bleak future of NAND flash memory. In Proceedings of the 10th USENIX conference on File and Storage Technologies (FAST '12). USENIX Association, 2012.
[28] Haryadi S. Gunawi, Cindy Rubio-González, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dussea, and Ben Liblit. EIO: Error Handling is Occasionally Correct. In Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST '08), pages 14:1–14:16, San Jose, CA, 2008.
[29] S Hur, J Lee, M Park, J Choi, K Park, K Kim, and K Kim. Effective program inhibition beyond 90nm NAND flash memories. Proc. NVSM, pages 44–45, 2004.
[30] Seok Jin Joo, Hea Jong Yang, Keum Hwan Noh, Hee Gee Lee, Won Sik Woo, Joo Yeop Lee, Min Kyu Lee, Won Yol Choi, Kyoung Pil Hwang, Hyoung Seok Kim, et al. Abnormal disturbance mechanism of sub100 nm NAND flash memory. Japanese Journal of Applied Physics, 45(8R):6210, 2006.
[31] Myoungsoo Jung and Mahmut Kandemir. Revisiting Widely Held SSD Expectations and Rethinking Systemlevel Implications. In Proceedings of the 2013 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '13), pages 203–216, 2013.
[32] Harendra Kumar, Yuvraj Patel, Ram Kesavan, and Sumith Makam. High Performance Metadata Integrity Protection in the WAFL Copy-on-Write File System. In Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST '17), pages 197–212, Santa Clara, CA, 2017. USENIX Association.
[33] Changman Lee, Dongho Sim, Jooyoung Hwang, and Sangyeun Cho. F2FS: A New File System for Flash Storage. In Proceedings of the 13th USENIX Conference on File and Storage Technologies (FAST '15), pages 273– 286, Santa Clara, CA, 2015. USENIX Association.
[34] Jae-Duk Lee, Chi-Kyung Lee, Myung-Won Lee, HanSoo Kim, Kyu-Charn Park, and Won-Seong Lee. A new programming disturbance phenomenon in NAND flash memory by source/drain hot-electrons generated by GIDL current. In Non-Volatile Semiconductor Memory Workshop, 2006. IEEE NVSMW 2006. 21st, pages 31–33. IEEE, 2006.
[35] Ren-Shuo Liu, Chia-Lin Yang, and Wei Wu. Optimizing NAND flash-based SSDs via retention relaxation. In Proceedings of the 10th USENIX conference on File and Storage Technologies (FAST '12), page 11, San Jose, CA, 2012. USENIX Association.

[36] Yixin Luo, Saugata Ghose, Yu Cai, Erich F Haratsch, and Onur Mutlu. HeatWatch: Improving 3D NAND Flash Memory Device Reliability by Exploiting SelfRecovery and Temperature Awareness. In 24th International Symposium on High Performance Computer Architecture (HPCA), pages 504–517. IEEE, 2018.
[37] Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, and Onur Mutlu. Improving 3D NAND Flash Memory Lifetime by Tolerating Early Retention Loss and Process Variation. Proceedings of the 2018 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '18), 2(3):37:1–37:48, December 2018.
[38] Avantika Mathur, Mingming Cao, Suparna Bhattacharya, Andreas Dilger, Alex Tomas, and Laurent Vivier. The New ext4 Filesystem: Current Status and Future Plans. In Proceedings of the Linux symposium, volume 2, pages 21–33, 2007.
[39] Justin Meza, Qiang Wu, Sanjev Kumar, and Onur Mutlu. A Large-Scale Study of Flash Memory Failures in the Field. In Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '15), pages 177–190, 2015. 796 2019 USENIX Annual Technical Conference USENIX Association
[40] Neal Mielke, Hanmant P Belgal, Albert Fazio, Qingru Meng, and Nick Righos. Recovery Effects in the Distributed Cycling of Flash Memories. In Proceedings of the 44th Annual International Reliability Physics Symposium, pages 29–35. IEEE, 2006.
[41] Neal Mielke, Todd Marquart, Ning Wu, Jeff Kessenich, Hanmant Belgal, Eric Schares, Falgun Trivedi, Evan Goodness, and Leland R Nevill. Bit error rate in NAND flash memories. In Proceedings of the 46th Annual International Reliability Physics Symposium, pages 9– 19. IEEE, 2008.
[42] Keshava Munegowda, GT Raju, and Veera Manikandan Raju. Evaluation of file systems for solid state drives. In Proceedings of the Second International Conference on Emerging Research in Computing, Information, Communication and Applications, pages 342–348, 2014.
[43] Iyswarya Narayanan, Di Wang, Myeongjae Jeon, Bikash Sharma, Laura Caulfield, Anand Sivasubramaniam, Ben Cutler, Jie Liu, Badriddine Khessib, and Kushagra Vaid. SSD Failures in Datacenters: What? When? And Why? In Proceedings of the 9th ACM International on Systems and Storage Conference (SYSTOR '16), pages 7:1–7:11, 2016.
[44] Nikolaos Papandreou, Thomas Parnell, Haralampos Pozidis, Thomas Mittelholzer, Evangelos Eleftheriou, Charles Camp, Thomas Griffin, Gary Tressler, and Andrew Walls. Using Adaptive Read Voltage Thresholds to Enhance the Reliability of MLC NAND Flash Memory Systems. In Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI (GLSVLSI '14), pages 151–156, 2014.
[45] Vijayan Prabhakaran, Lakshmi N. Bairavasundaram, Nitin Agrawal, Haryadi S. Gunawi, Andrea C. ArpaciDusseau, and Remzi H. Arpaci-Dusseau. IRON File Systems. In Proceedings of the Twentieth ACM Symposium on Operating Systems Principles (SOSP '05), pages 206–220, Brighton, United Kingdom, 2005.
[46] Ohad Rodeh, Josef Bacik, and Chris Mason. BTRFS: The Linux B-Tree Filesystem. ACM Transactions on Storage (TOS), 9(3):1–32, August 2013.
[47] Marco AA Sanvido, Frank R Chu, Anand Kulkarni, and Robert Selinger. NAND flash memory and its role in storage architectures. Proceedings of the IEEE, 96(11):1864–1874, 2008.
[48] Bianca Schroeder, Raghav Lagisetty, and Arif Merchant. Flash Reliability in Production: The Expected and the Unexpected. In Proceedings of the 14th USENIX Conference on File

and Storage Technologies (FAST '16), pages 67–80, Santa Clara, CA, 2016. USENIX Association.
[49] Kang-Deog Suh, Byung-Hoon Suh, Young-Ho Lim, JinKi Kim, Young-Joon Choi, Yong-Nam Koh, Sung-Soo Lee, Suk-Chon Kwon, Byung-Soon Choi, Jin-Sun Yum, et al. A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme. IEEE Journal of Solid-State Circuits, 30(11):1149–1156, 1995.
[50] Hung-Wei Tseng, Laura Grupp, and Steven Swanson. Understanding the Impact of Power Loss on Flash Memory. In Proceedings of the 48th Design Automation Conference (DAC '11), pages 35–40, San Diego, CA, 2011.
[51] Yongkun Wang, Kazuo Goda, Miyuki Nakano, and Masaru Kitsuregawa. Early experience and evaluation of file systems on SSD with database applications. In 5th International Conference on Networking, Architecture, and Storage (NAS), pages 467–476. IEEE, 2010.
[52] Mai Zheng, Joseph Tucek, Feng Qin, and Mark Lillibridge. Understanding the Robustness of SSDs Under Power Fault. In Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST '13), pages 271–284, San Jose, CA, 2013. USENIX Association.
[53] Mai Zheng, Joseph Tucek, Feng Qin, Mark Lillibridge, Bill W. Zhao, and Elizabeth S. Yang. Reliability Analysis of SSDs Under Power Fault. ACM Transactions on Storage (TOS), 34(4):1–28, November 2016.