学号 2017202110042
密级

海量存储技术结课论文

降低基于 NAND 闪存的固态硬盘的读写延迟方法调研

院（系）名 称 ： 计算机学院

专 业 名 称 ： 计算机软件与理论

学 生 姓 名 ： 廖庆文

学 号 ： 2017202110042

二〇一七年十二月

郑 重 声 明

本人呈交的学位论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本学位论文的知识产权归属于培养单位。

本人签名：_____ 日期：_____

摘　要

固态硬盘（SSD）是一种以内存作为永久性存储器的存储设备，在接口的规范和定义、功能及使用方法上与普通硬盘的完全相同，在产品外形和尺寸上也完全与普通硬盘一致，因此被广泛应用于军事、车载、工控、视频监控、网络监控、网络终端、电力、医疗、航空、导航设备等领域。现在 SSD 已不是使用“硬盘”来记存数据，而是使用 NAND Flash。闪存（Flash）在访问延迟、传输带宽、价格方面介于 DRAM 和磁盘之间，而且密度比 DRAM 和磁盘高，能耗比它们低，因而闪存成为了固态硬盘的主流存储介质。基于闪存的固态硬盘的存储技术一度成为研究的热点。本文总结了目前基于闪存的固态硬盘的存储技术，重点介绍了两个已有的相关工作在提升 SSD 的存储性能方面提出的方法，以及与其相关的研究问题和关键技术，并且基于对研究现状的分析，提出了一点看法。

关键词：固态硬盘；闪存；存储技术

ABSTRACT

Solid State Drive (SSD) is a memory device that uses permanent memory as a storage device. It has the same specifications and definitions, functions, and usage as the ordinary hard disk drive. It is also completely consistent with the ordinary hard disk drive in terms of product form factor and size. It is widely used in military, automotive, industrial control, video surveillance, network monitoring, network terminals, power, medical, aviation, navigation equipment and other fields. Now that SSDs do not use "hard drives" to store data, they use NAND Flash. Flash memory has become the mainstream storage medium for solid state drives because of its access delay, transmission bandwidth, and price, between DRAM and disk, and higher density than DRAM and disk. The storage technology of flash-based SSDs has been the hotspot of research. This paper summarizes the current flash-based storage technologies for SSDs, highlighting the two existing work related to improving the storage performance of SSDs, as well as their associated research issues and key technologies. And based on the analysis of the status quo, made a point of view.

Key Words: Solid State Disk; flash memory; storage technique

目　录

1 绪论

闪存是一种电可擦除可编程只读非易失性存储器，也叫 Flash EEPROM1，最初于 1984 年由东芝公司发明。早期由于价格昂贵，闪存最先被应用于嵌入式系统作为标准存储器。随着器件制作密度的提高和价格的降低，闪存被应用到笔记本电脑中代替磁盘，以及被应用到企业级存储的高端存储阵列中。目前随着大规模数据中心的电力和相关的冷却开销成为制约数据中心密度和能力增长越来愈重要的因素，低能耗、高性能的闪存已被引入到以数据处理为中心（Data-Intensive Computing）的高性能计算系统中[12]。

固态硬盘通过内部的闪存转换层，对外提供块设备访问接口，对已有的文件系统和应用程序完全兼容，减少了开发和测试新软件的代价，是闪存技术能迅速得到广泛应用的重要因素之一。由于与传统存储体系结构兼容，目前固态硬盘的应用集中在个人电脑、数据中心等场景中，用以全部或部分代替磁盘作为系统的外存储设备。闪存转换层作为固态硬盘的关键技术，成为研究重点。

闪存的应用将越来越广泛，目前闪存已成为存储用户的主流目标。而固态硬盘消费市场继续以显著的速度增长，SSD 支持的云虚拟主机实例正在成为常态，闪存/ SSD 阵列已成为高端存储服务器中流行的解决方案。

从用户方面来说，他们有着快速和延迟稳定的要求。然而，用闪存作为存储单元的 SSD 并不能总是为用户提供期望的性能。人们发现很难通过 SSD 来满足服务级别的需求，并且在部署 SSD 后的 700 万小时中发现了性能高低的显著变化[7]。这主要是由于闪存有结构上的缺点：每个存储单元只有在擦除以后才能写数据，并且擦除操作所需的时间比写操作多一个数量级；另外每个存储单元的擦除次数有限，而且随着密度增高，可擦除次数减少。这些缺点在应用闪存时不可避免，因此如何克服这些缺点，提升 SSD 的读写性能已经成为目前研究的一个热点。

本文首先介绍了基于闪存的固态硬盘技术，着重介绍了 SSD 中的地址映射过程和垃圾回收机制以及擦写均衡。随后重点论述了提升固态硬盘读写性能的研究现状，深入分析了近几年来国内外发表的两个具有代表性的研究成果。针对上文提到的问题和挑战，本文介绍的两篇论文的侧重点，分别是从暂停擦除任务和减少垃圾回收的影响两个方面探讨了提升 SSD 读写性能的方法。最后提出了当下面临的挑战和今后的工作方向。

2 基于闪存的固态硬盘技术

SSD 的逻辑结构如下图 2.1 所示，由以下几个部分组成：1）接口，是 SSD 与计算机之间的通道，目前主要有 SAS、SATA 和 PCI-E 接口，SATA 和 SAS 接口的吞吐率是 3Gbps 和 6Gbps，而 PCI-E 接口的吞吐率更高。通过这些标准接口 SSD 很好地隐藏了其复杂的设计。2）Flash Package,是闪存部件，负责数据存储和读写。SSD 中一般有多个 NAND Flash，每个 NAND Flash 包含多个 Block，每个 Block 包含多个 Page。由于 NAND 的特性，其存取都必须以 page 为单位，即每次读写至少是一个 page，通常地，每个 page 的大小为 4k 或者 8k，其结构如图 2.2 所示。3）SSD 控制器是 SSD 的核心，控制器实现磁盘页和 SSD 页面的映射、SSD 空间管理、block 擦写均衡等等。控制器内部包括处理器、缓存管理模块和 Flash demux。缓存管理模块，管理发往 SSD 和命令和数据；Flash demux 负责控制多个 Flash package。4）内存。存储和缓存各类数据。

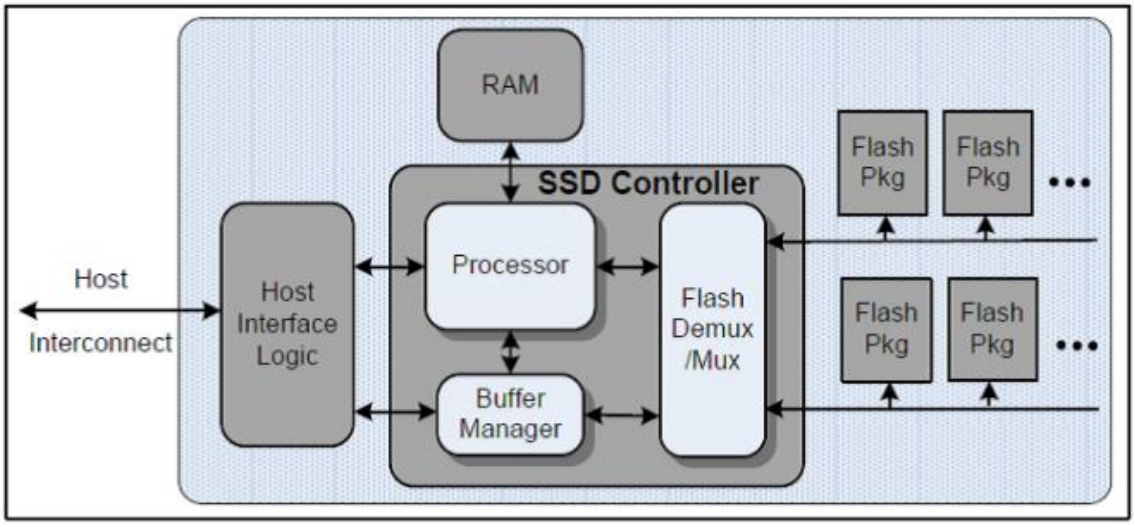

图 2.1　SSD 内部结构示意

地址映射、垃圾清理、擦写均衡是 SSD 控制器的三大功能。由于 SSD 不支持页面覆盖操作，每次写入时，逻辑页面将会映射到不同的物理页面，因此必须在逻辑地址（LBA）和物理地址（PBA）之间建立映射关系。地址映射的粒度可以是页面或者块，页面级别映射更加灵活高效，但是需要占用更多的存储空间，块级别的映射虽然占用存储空间少，但会导致更多的 read-modify-write 操作，也就是写放大。两种方法各有优缺点，因此也有人提出多粒度映射的方法，大部分数据是按照块映射，小部分最新写入的数据按照页面映射。除此之外，还有混合地址映射和变长式

地址映射。混合式地址映射机制指的是对频繁更新的数据维护基于页的地址映射表，而对大量的其他数据维护基于块的地址映射表。

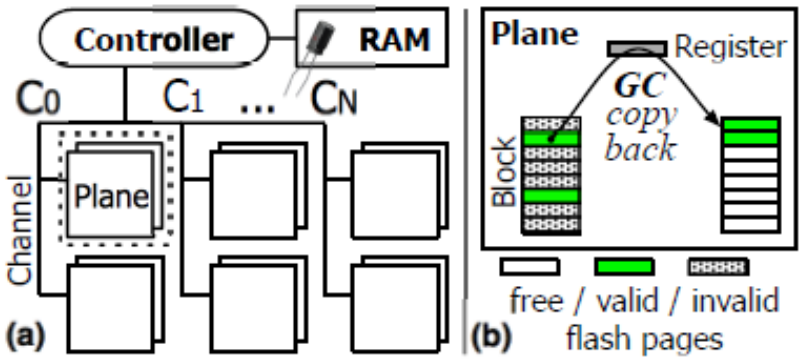

图 2.2 plane 连接示意即内部结构图

由于 SSD 的块必须先擦除再写入，这主要收到了 NAND 闪存的特性的限制，一个是 NAND Flash 每次写必须以 page 为单位，且只能写入空闲的 page，不能覆盖写原先有内容的 page 擦除数据时，由于电压较高，只能以 block 为单位。擦除 SSD 内部一般预留一部分空间（over-provision）。页面写入操作的过程非常类似于写日志：先将页面追加到空闲空间中，之后更新页面映射关系。只要 SSD 中有足够的空闲页面，写操作不会导致写放大。当空闲空间不足时，SSD 将启动垃圾回收算法，垃圾回收的目的是将有效地页面集中起来，释放无效页面占用的空间。

垃圾回收过程首先选择一个块作为回收对象，将其中的有效页复制到一个空闲块中，然后更新地址映射表，擦除该块，最后把它加入空闲块列表中。由于垃圾回收的过程涉及数据复制、擦除等操作，垃圾回收机制的优劣影响固态硬盘的读写性能和可靠性，因此成为固态硬盘性能瓶颈。垃圾回收对读写性能造成的影响也是 SSD 新盘写入速度较高，当数据量增大时读写性能下降的原因之一。

Flash 是有擦写次数上限的，为了延长寿命，SSD 盘需要擦写均衡（wear-leveling）功能，即尽量让每个块的擦写次数是均衡的，擦写均衡算法较多且复杂。由于垃圾回收过程中涉及到擦除操作，磨损均衡是垃圾回收算法需要考虑的重要方面。但如果仅在垃圾回收过程中考虑，无法在闪存芯片上所有的块范围内实现磨损均衡。因此还需要设计全局的磨损均衡算法，全局的磨损均衡算法的主要思想是每隔一定时间，将闪存存储器上的冷、热数据进行交换以达到磨损均衡的目的。

3 研究现状分析

从上一节的描述可以发现，在 NAND 闪存中，如果向一个无效的页写入数据的话，必须要先对这个页进行擦除才能用于写入，一旦向 NAND 闪存芯片发出页面编程或块擦除（P / E）命令，随后的读取请求必须等待直到耗费时间的 P / E 操作完成。初步结果显示，长时间的 P / E 操作可能会使读取延迟平均增加 2 倍。而又由于垃圾回收的过程涉及数据复制、擦除等操作，即多次 P/E 操作，这就对 SSD 的读取造成了很大的影响。

3.1 通过挂起 NAND 闪存的 P/E 操作降低 SSD 读取延迟

由于 NAND 闪存的 P / E 速度较慢，一旦 P / E 被提交给闪存芯片，等待或随后的读请求遭受等待时间导致的延长的服务延迟。由于磁盘读取请求是由高级别高速缓存未命中导致的，因此磁盘的读取延迟时间会降低应用程序的性能。为了解决 P/E 操作与读取的竞争问题，已有的相关工作包括在驱动器的空闲时间执行垃圾回收，这种方法只能缓解这种竞争问题，并不能真正的解决。此外，读取请求可以在待处理列表中优先化，以减少由 P / E 引起的排队时间。

在前人研究工作的基础上，Wu 等人在 2012 年时提出了一种低开销的 P / E 挂起方案[2]，这种方案的提出是受到了 P/E 算法内部机制的启发，其主要的思想是暂停正在执行的 P/E 操作以便为待处理的读取提供服务，在读取完成后，恢复被暂停的 P/E 操作[6]。下面具体介绍该方法的实现过程。

3.1.1 擦除的挂起和恢复

在 NAND 闪存中，擦除过程包括两个阶段：首先，在目标块上施加一个持续T_{erase}的擦除脉冲; 其次，执行采取T_{verify}的验证操作，以检查前面的擦除脉冲是否成功擦除了块中的所有位。否则，重复上述过程直到成功，或者如果迭代次数达到预定的限制，则报告操作失败。

擦除挂起。暂停擦除脉冲或验证操作需要重置连接闪存单元与模拟模块的相应导线的状态。具体而言，由于闪存在不同的操作中工作在不同的电压偏置状态下，所以需要为读取请求重新设置施加在导线的电流偏置电压。在擦除脉冲或者验证操作之后都要进行这个过程，记为$O_{p_{voltage_reset}}$，这个过程的时间花销为$T_{\text{voltage_reset}}$。

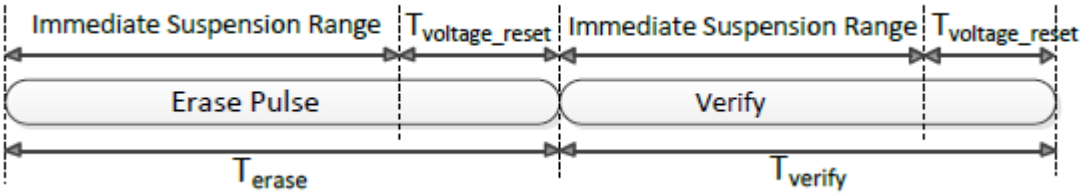

图 3.1　擦除操作时间表

因此，如果在$O_{p_{voltage_reset}}$期间挂起命令到达，$O_{p_{voltage_reset}}$完成后暂停将成功（如图 3.2）。否则，立即执行$O_{p_{voltage_reset}}$，然后由芯片提供读取请求（如图 3.3）。

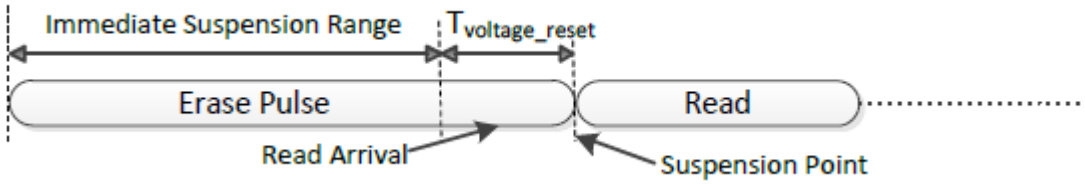

图 3.2

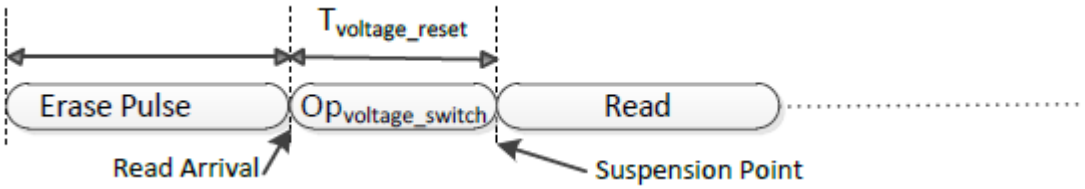

图 3.3

恢复擦除操作。恢复意味着 NAND 闪存的控制逻辑恢复暂停的擦除操作。 因此，控制逻辑应该跟踪进度，即在验证阶段或擦除脉冲期间是否发生暂停。如果在验证阶段暂停，验证操作必须重新完成。否则，如果没有更多的暂停发生，则在恢复过程中将完成剩余的擦除脉冲时间（T_{erase} – 已完成的部分擦除时间）。实际上，NAND 闪存的控制逻辑中现有的设备可以很容易地支持进度跟踪的任务：脉冲宽度发生器使用类似于计数器的逻辑来实现[13]，该逻辑跟踪当前脉冲的进度。

3.1.2 编程的挂起和恢复

要写入的数据通过控制器总线传输并加载到页面缓冲区中，然后执行 ISPP（增量步进脉冲编程）[4]，其中在目标闪存页面上执行由编程阶段和验证阶段组成的N_{w_cycle}次迭代。在每个 ISPP 迭代中，程序阶段负责在单元上施加所需的编程电压

偏置以给它们充电。在验证阶段，读取单元的内容以验证是否在每个单元中存储了期望的电荷量：如果是，则认为该单元是程序完成;否则，在该单元上再进行一次 ISPP 迭代。程序过程如图 3.4。

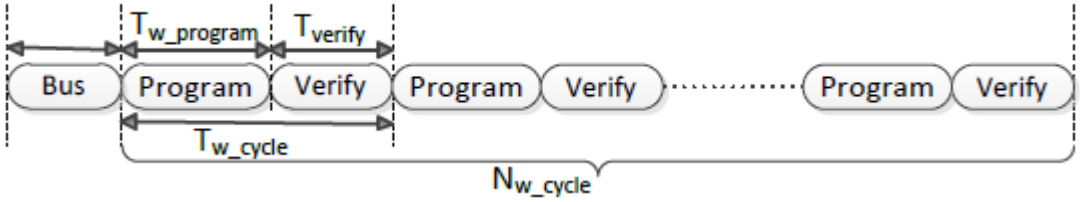

图 3.4

暂停需要考虑的一个最重要的问题就是保存页面缓冲区的内容。如果写入被读取抢占，页面缓冲区的内容肯定会被替换。 因此，写入的恢复要求页面缓冲器被重新存储。为了降低保存缓冲区页面数据副本的开销，Wu 等人提出了一个影子缓冲区，影子缓冲区就像页面缓冲区的副本一样，在写入请求到达时自动加载页面缓冲区的内容，并在恢复时重新存储页面缓冲区。影子缓冲区与页面缓冲区并行连接，因此它们之间的数据传输可以实时完成。

暂停编程操作。与擦除脉冲的长度（T_{erase}）相比，编程过程的编程和验证阶段通常缩短两个数量级。因此，Wu 等人针对编程操作的暂停，提出了两种策略，一种是“相间暂停”（IPS），直观地说，程序进程可以在任何 ISPP 迭代的程序阶段结束时以及验证阶段结束时暂停。另一种策略为“阶段内部取消”（IPC），类似于取消擦除暂停的验证阶段，IPC 取消正在进行的程序或在暂停时验证阶段。 取消暂停程序阶段的原因是程序阶段$T_{\text{w_program}}$的持续时间较短，通常被认为是原子的。对于 IPC，如果在编程或验证阶段进行$O_{p_{voltage_reset}}$时读取到达，则实际上在阶段结束时发生暂停，这与 IPS 相同。显然，IPS 在写入上的开销比 IPC 小，但读取性能相对较低。

恢复被暂停的编程操作。对于 IPS 策略的恢复，首先，页面缓冲区被重新加载影子缓冲区的内容。然后，控制逻辑检查最后的 ISPP 迭代次数和前一个阶段。如果 IPS 在验证阶段结束时发生，我们可以继续下一个 ISPP。另一方面，如果最后一个阶段是程序阶段，当然我们需要在进行下一个 ISPP 迭代之前完成验证操作。恢复过程如图 3.5。

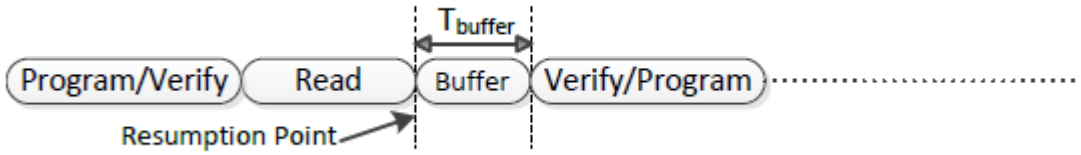

图 3.5 IPS 策略暂停编程的恢复过程

对于 IPC 策略的恢复，与 IPS 相比更为复杂。编程操作将电荷放入单元中，从而改变单元的阈值电压（V_{th}）。因此，我们需要通过验证操作来确定取消的编程阶段是否已经达到期望的V_{th}（即，数据是否可以被认为写入单元中）。如果是这样，在这个单元上不需要更多的 ISPP 迭代，否则，先前的程序操作再次在单元上执行。恢复过程如图 3.6。

图 3.6 IPC 策略暂停编程的恢复过程

3.1.3 实现结果

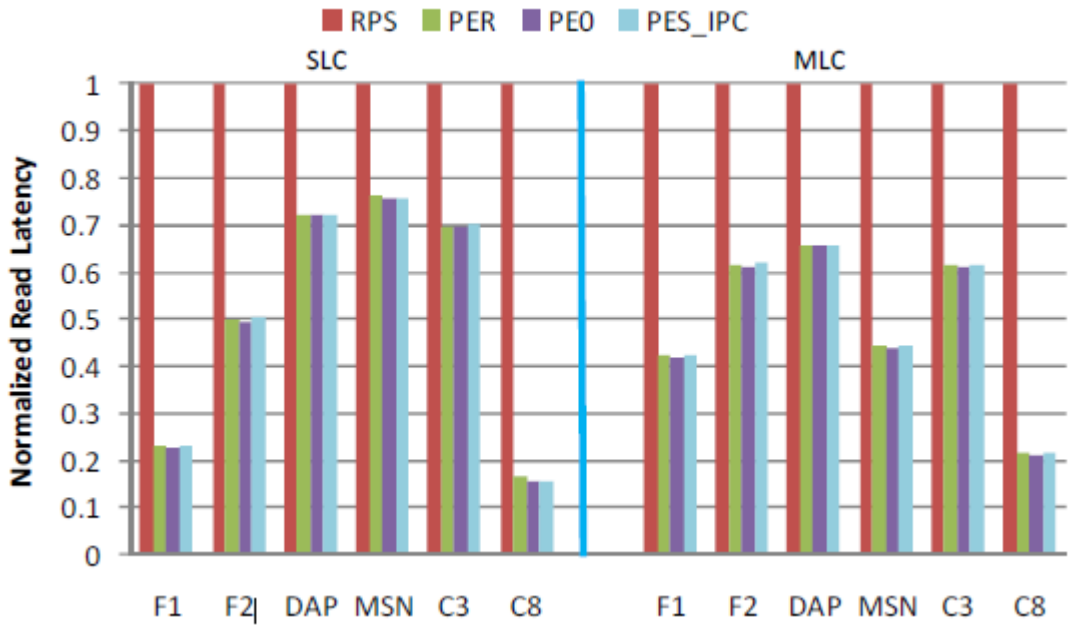

图 3.7 四种情况平均读延迟

实验中，Wu 等人首先比较 P / E 挂起策略与 RPS，PER 和 PE0 三种方法的平均读取延迟进行比较，其中结果被归一化为 RPS 的结果，如图 3.7 所示。对于 P / E 挂起，在图 3.7 中采用挂起 IPC 策略，表示为“PES_IPC”。将编程和擦除的物理

等待时间值设置为零的 PE0 用作乐观情况，其中读取与 P / E 之间的争用被完全消除。同理，PER 表示将编程和擦除的延迟设置为等于读取的延迟的情况。图 2 表明，与 RPS 相比，所提出的 P / E 暂停实现了显着的读取性能增益，这与最佳情况 PE0 几乎相同，其差值小于 1%。具体而言，PES_IPC 平均 6 条曲线的平均读取延迟时间为 SLC 48.9%，MLC 50.5%，RLC 为 71.6%，MLC 为 75.4%。

与 IPS 相比，IPC 可以实现更好的读取性能，但会导致更高的写入开销。图 3.8 比较了 IPC 和 IPS 的读取性能，IPS 的读取延迟平均为 8.0%和 2.7%，最高为 13.2%和 6.7%，高于 IPC。这是因为 IPS 具有额外的读取延迟，这个额外的延迟是读取请求到达和程序结束或验证阶段的暂停点之间的时间。

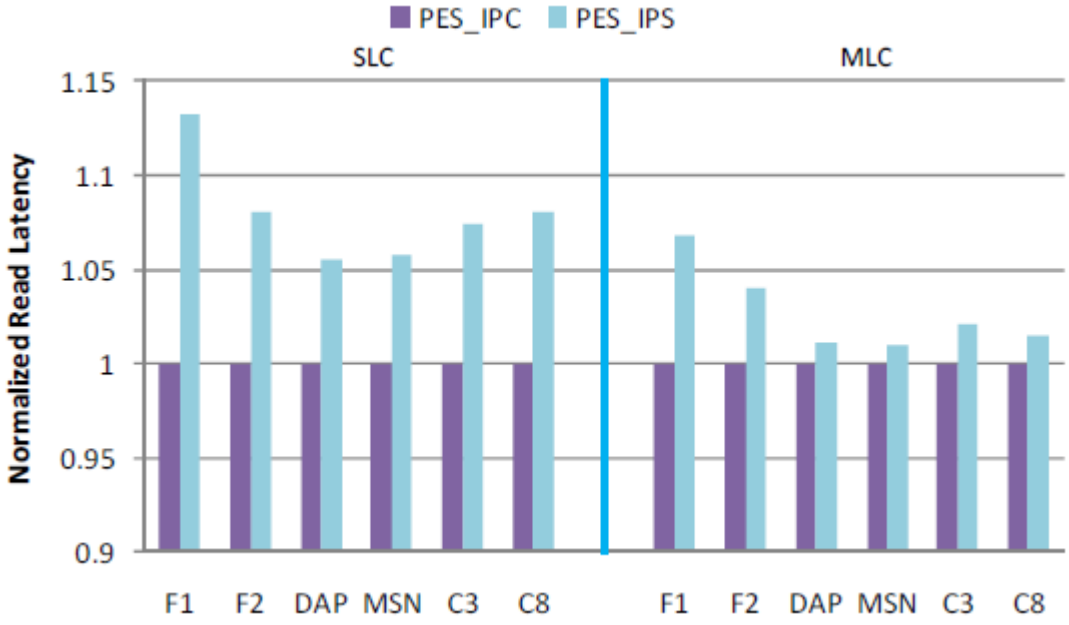

图 3.8 IPS 和 IPC 策略的平均读延迟

3.1.4 小结

如上文所述，NAND 闪存的一个性能问题是其编程和擦除延迟远高于读取延迟，这个问题导致读取和 P / E 之间的竞争，如果每次读取都要等待当前的 P/E 完成，那读取性能将会受到很大的影响。如果使用当前的 NAND 闪存接口，正在进行的 P / E 不能被暂停和恢复。为缓解芯片竞争对读取性能的影响，Wu 等人利用 NAND 闪存中 P / E 算法的内部机制，提出了一种轻载开销的 P / E 暂停方案。通过精确的时序和多芯片/通道的现实 SSD 建模来模拟/评估设计。并在最后的实验中给出了证明，他们在引入额外的微不足道的写入开销的基础上提出的 P / E 暂停方法显著地降低了读取延迟。

3.2 消除 NAND SSD 中垃圾回收的尾部延迟

导致闪存性能不稳定的核心问题是众所周知的垃圾收集（GC）过程，正如第二章中提到的，垃圾回收机制成为固态硬盘性能瓶颈，其优劣影响固态硬盘的读写性能和可靠性。

具体的，当发生垃圾回收时，控制器会在每一个页面的拷贝操作中，循环执行下述的四个步骤：

1. 通过通道（耗时仅 0.2μs）发送一个“flash-to-register”的读取命令到正在执行 GC 的闪存面。

2. 等待，直到闪存面开始执行“1-page”的读取命令（约 40μs 不使用通道）。

3. 发送一个“flash-to-register”的写入命令。

4. 等待，直到闪存面开始执行“1-page”的写入命令（约 800μs 不使用通道）。

当 GC 活动使用某些资源（例如，控制器，通道，闪存面）时会发生 GC 阻塞，这将会延迟后续的请求，使用阻塞的设计主要是因为它们简单又便宜的。 然而，由于 GC 延迟时间过长，在 GC 过程中 SSD 无法处理（或阻止）传入的 I/O，阻塞式设计会导致出现明显的尾部延迟。受到正在进行的垃圾回收的影响，读取延迟差异可能达到 100 倍的水平。垃圾回收的阻塞实现又分为两种，一个简单方法就是使用阻塞控制器。也就是说，即使只有一个闪存面在执行 GC 操作，控制器就会因为忙于与该闪存面进行通信，而无法处理指派给其他闪存面的未完成 I/O，我们将其称之为控制器阻塞型 GC。另一种方法是使用带通道队列功能的多线程/多 CPU。当某个线程或 CPU 正与垃圾回收的平面通信并阻塞该 plane 对应通道的时候，其他线程或 CPU 可以处理指派给其他通道的 I/O 操作。我们将其称之为通道阻塞型 GC。

受到上述问题的启发，各种针对垃圾回收机制进行改进以提高固态硬盘读写性能方法随之而来，其中，Yan 等人就提出了一种“小尾巴”闪存的方法，这种方法能够几乎完美的消除 SSD 中由于垃圾回收机制造成的尾部延迟，从而有效的提升了读和写的速度。与传统的致力于减少垃圾回收发生的次数的各类技术不同，Yan 等人提出的方法试图通过消除垃圾回收操作的阻塞性[1]，来为长期运行的 SSD 提供稳定性能。

目前，SSD 内部技术已经发生了很多方面的变化。首先，闪存控制器的功能和速度与以前相比有了很大的提升，在控制器上实现了如多线程，I/O 并发，细粒度

的 I/O 管理等更为复杂的逻辑。其次，现代固态硬盘的误码率已经上升到 ECC 纠错码难以应付的成都，由于疯涨的误码率，现代商业固态硬盘采用了基于奇偶校验的冗余阵列（RAIN）来作为标准的数据保护机制。Yan 等人有效的利用了这种独立的 NAND 冗余阵列（RAIN），通过奇偶校验的再生机制，来规避垃圾回收带来的阻塞以读取 I / O。最后，现代固态硬盘还配备了一个由“超级电容”构成的大型 RAM 缓冲区，利用这些缓冲区可以解决垃圾回收操作中的写入尾部延迟问题。

得益于上述 SSD 的新技术，Yan 等人由此提出了四种新的调度策略来消除垃圾回收操作的阻塞性问题，从而有效的提高了 SSD 的性能。这四种策略分别是：plane-blocking GC、GC-tolerant read、rotating GC、GC-tolerant flush。

3.2.1 plane-blocking GC

为了降低这些控制器阻塞和通道阻塞等不必要的阻塞，设计了一个无阻塞的控制器和通道协议，将任何阻塞的资源从 GC 推送到受影响的闪存面。我们称之为细粒度架构的 plane-blocking GC。也就是说，只有正在 GC 过程的平面中发生了 I/O 阻塞。其他所有没进行 GC 的平面中的 I/O 正常运行，包括了发生 GC 的平面在同一通道中的平面。在控制器 cpu/线程发送一个 flash-to-register 的读/写指令后，控制器创建一个未来事件以标记完成时间。控制器可以可靠的预测在平面内的读/写指令的耗时，接着转而去处理其他 I/O 操作。这样，在一个有 GC 操作的 plane 内的回拷的持续期间，控制器可以持续的为该通道的其他没有进行 GC 操作的 plane 提供 I/O 服务。这是利用了 SSD 的新技术——平面内回拷技术实现的。

3.2.2 GC- tolerant read

为了解决正在进行垃圾回收的平面的 I/O 阻塞问题，我们采用了 RAIN 技术。根据前面描述的知识，每个 SSD 内部都有很多个通道，每个通道又会连着多个平面。为了解决上述问题，在该策略的思想下，每个通道内，在相同位置的平面组成一个平面组，每一个组设置一个平面用于奇偶校验。如图 3.9 所示的P_{012}, P_{345}, P_{678}就是用于奇偶校验的平面。如果一个分页因为正在进行的垃圾回收不能读取，分页内容将迅速通过读取另一个平面上的校验分页内容来进行再生成。

为了防治校验通道瓶颈，Yan 等人在此基础上还加入了一些小改动，把各个奇偶校验平面分散到不同的通道中，呈对角线分布。实现这样的策略的一个重要的基础是，Yan 等人创建的一个静态-动态混合映射方式。

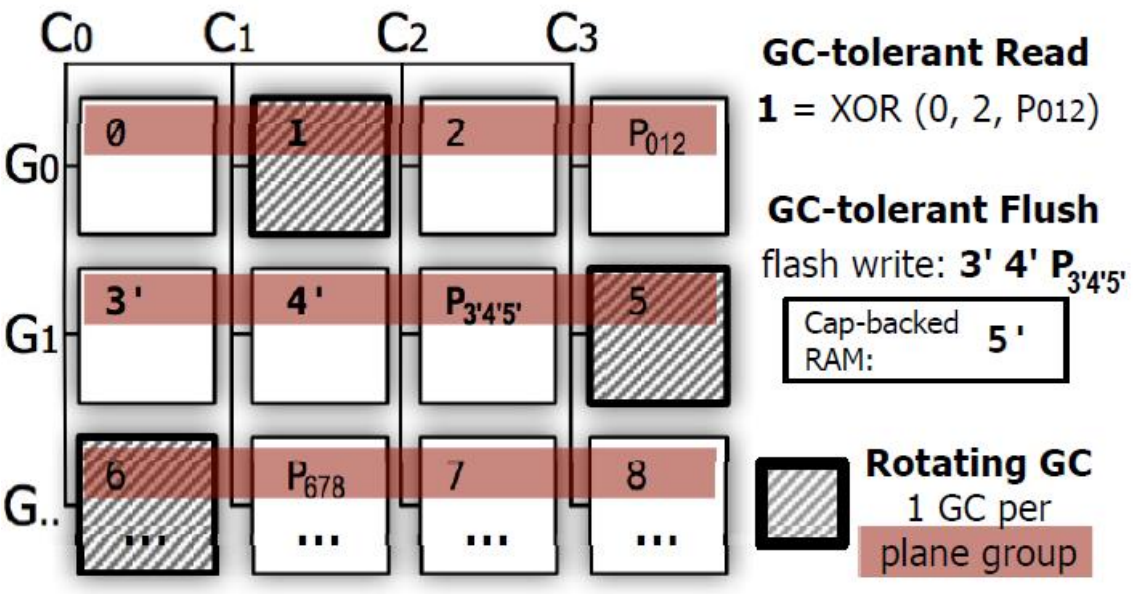

图 3.9 平面组示意图

3.2.3 rotating GC

当一个 plane 组中多个 plane 同时进行 GC 操作时，上一种策略则无法工作，因为一个校验页只能切断一条“尾巴”，当多个页面同时进行 GC 的时候，这些页上的内容将无法被计算出来。由此 Yan 等人提出了 rotating GC 策略，强制在同一时间一个 plane 组中最多只能有一个 plane 可以进行 GC 操作。在不同 plane 组的 GC 操作仍然可以并发运行。rotating GC 的实现依赖于 RAIN 的架构所确保的每一条带静态的映射到对应的 plane 组。但该策略使用时要考虑通道的个数造成的影响，如果使用大的条带宽，不仅违背了 rotating GC 策略，还因为降低了可靠性以及局部读取在 GTR 下会造成更多的额外 I/O 生成，所以在多通道的 SSD 中，尽量保证小一点的条带宽。

3.2.4 GC-Tolerant Flush

上述三种策略仅解决了读取的尾延迟，对于写入来说，这个过程更为复杂，为了处理这种复杂的问题，Yan 等人使用了工业界大量采用的基于电容器的 RAM 作为持久写入的缓存，当缓存占比超过 80%，一个后台刷新程序将会回收一些分页。当缓存满了时，一个前台刷新程序将会运行，将会阻塞住写入直到有空闲分页为止。而 GC-tolerant flush 确保了分页回收与 GC 阻塞无关。对于全条带写来说，通过使用策略 1 和策略 3，每一个条带上将只有一个平面会进行 GC，并且同一条带上的其他平面的读写不会被整个 GC 的平面阻塞。而对于部分写，则先根据策略 2 读出进行垃圾回收的平面的内容，然后写入新的数据，最后再根据这 N-1 个平面内

容生成新的奇偶校验页，这样就完成了一次写入。

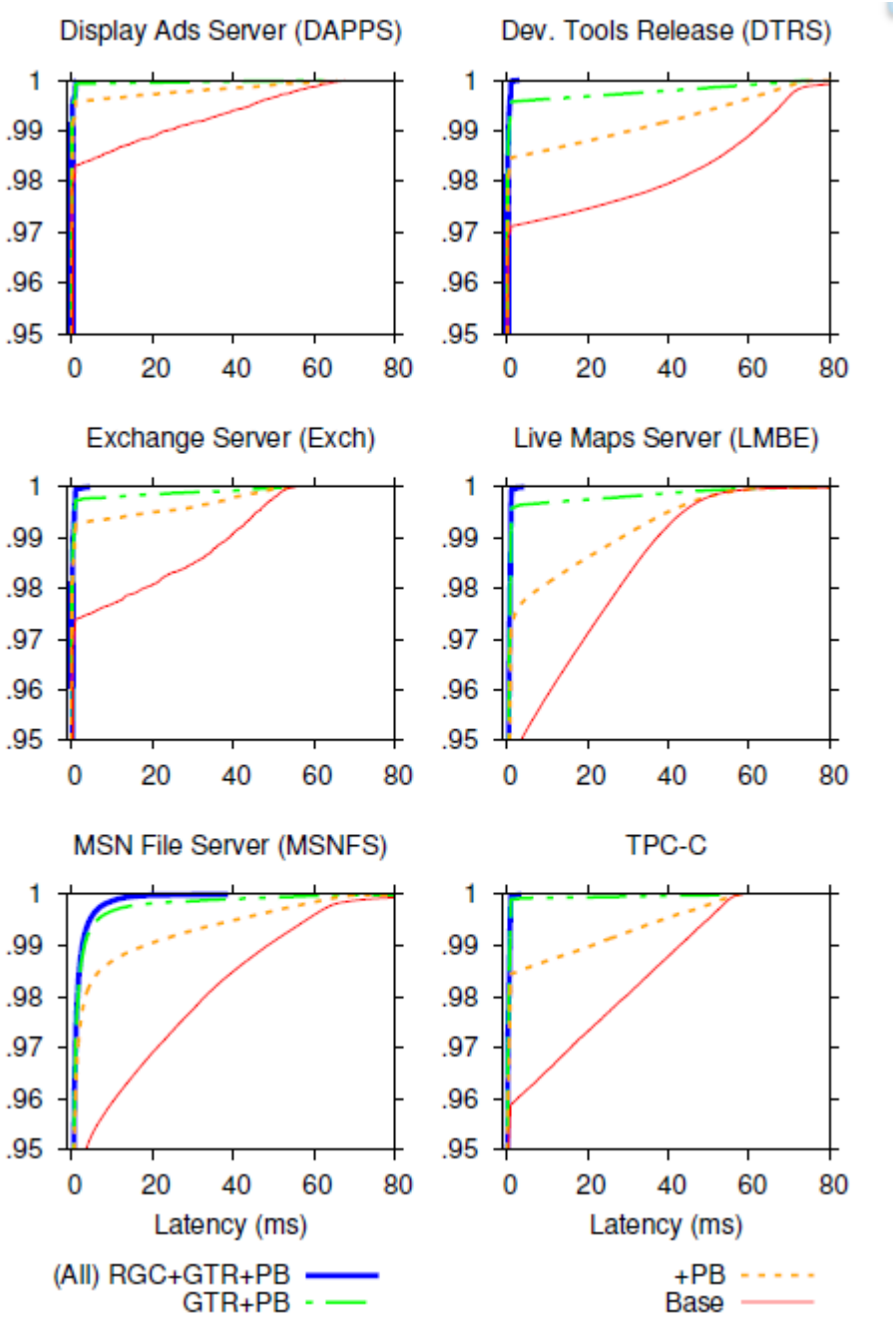

图 3.10 尾部延迟

3.2.5 实验结果

图 3.10 显示了在 TTFLASH-SIM 上运行的六个跟踪驱动实验的 CDF（Contiguous-Disk File）的读取延迟。这里只显示读取延迟。由于有 RAM 缓存，写入延迟快速而稳定，所以可以不进行考虑。如图 3.10 所示，基本方法（“BASE”=具有通道阻塞和最佳 FT 且没有 RAIN（固态硬盘阵列）的默认 SSDSim）具有较长的尾部延迟。相比之下，在逐个添加每个 TTFLASH 特征之后：+ PB（§3.2.1）

（plane-blocking），+ GTR（§3.2.2）(GC-Tolerant Read)和+ RGC（§3.2.3）(Rotating GC)，实验结果改进显著。当所有的特征被添加（RGC + GTR + PB）时，尾部的微小延迟接近无 GC 的情况。

3.2.6 小结

近年来 SSD 技术飞速发展，为了能执行复杂逻辑，许多更快更强大的 flash 控制器逐渐被生产出来；基于奇偶校验的 RAIN 已经成为一种成熟的数据保护的标准手段；而电容支持的 RAM 是针对低效写的一种解决方案。在 Yan 等人的工作中，开创地结合这些技术，提出了一些新方法，如：plane-blocking GC, rotating GC, GC-tolerant，read and flush。它们共同提出了解决 GC 导致的读写尾延迟的方案。

4 总结与展望

4.1 总结

本文详细介绍两种解决 SSD 中读写延迟的方案。从解决问题的基本着落点上来看，Wu 等人和 Yan 等人的工作都是从读或写操作不应被当前 SSD 内部正在进行的一些行为如垃圾回收影响为出发点的，从而提出各自的解决方案使得读取请求能够立马执行而不需等待或不需等待太久，通过这样的方式来提高 SSD 的读取性能。他们二人的工作与很多已有的方法不同。据了解，为了解决垃圾回收对闪存性能的影响，在过去十多年里，已经有了大量的通过各类新技术来减少垃圾回收次数的研究工作[3,8]，而几乎没有相关工作是从消除垃圾回收或垃圾回收中的擦除操作造成的阻塞来提升 SSD 性能的。由此可以看出，本文介绍的两个相关的研究工作是十分具有创新性的。

更具体的，Wu 等人的工作着力于解决 NAND 闪存芯片的编程/擦除操作和读取命令的竞争问题，当读取命令到达时，允许暂停正在进行的编程/擦除操作，以便为待处理的读取提供服务，从而解决了读取需要经过长时间的等待的问题，这种思想类似于在[9]的相关工作中就体现出来的将操作分解为小的阶段来抢占低优先级操作的思想。但[9]的工作中没有讨论每次编程/验证的迭代阶段可能有两个暂停点的问题，也没有解决恢复被暂停的操作时重新传输写入数据的开销。在 Wu 等人的工作中，提出了影子缓冲区来很好的解决了这个问题。

在 2017 年的 fast 会议上 Yan 等人发表的工作，立意比较新颖。基于上述 Wu 等人推迟当前操作的思想，KIM 等人和 AGRAWAL 等人都分别采取了在驱动器的空闲时间执行垃圾收集过程也可以缓解读取延迟的问题[10,11]，但这种方法不能从根本上解决问题，如果当前 SSD 已经没有空闲界面时，则垃圾回收会被立马执行而不是等到空闲时间，当这样的情况发生时，还是读写操作还是会产生一个尾部延迟。所以 Yan 等人提出了一种较为特别的方法，没有从解决垃圾回收和读写操作之间的争用的方面来考虑，而是提出了四种策略能够使得垃圾回收和读写操作能够并发的执行，这样，就基本能消除了 GC 操作引起的尾部延迟，使得一个使用了很久的 SSD 的读写性能尽可能的和新的 SSD 的读写性能接近。

4.2 展望

在学习了这两种相关的研究工作之后，我对此提出了一些个人的看法。除了上述已经提到的其他学者的相关工作的不足之处之外，Wu 和 Yan 的工作也存在着一些相应的问题。

首先，对于 Wu 的工作来说，如果当发生大量的读请求时，那么原来的编程/写入操作就会被不停的挂起和恢复。由于 SSD 不能覆盖写，必须要先擦除这些无效块才能写入，这时就会对闪存发出编程/擦除（P/E）命令。如果在 P/E 过程中产生了大量的读取请求，那么这样的写入操作就会由于 P/E 被挂起而一直推迟，对写入操作的性能会产生一个很大的影响。

对于 Yan 的工作，由于该方案依靠 RAIN，所以每 N 个通道中要浪费 1 个通道。增大 N 会减少浪费的通道，但写操作时的尾延时会增加。其次，在频繁的写操作时不能消除所有的尾延时。最后，Yan 等人提出的 TTFlash 采取的是平面内回拷的方法，所以跳过了 ECC 检查。

针对上述的这些问题，我认为，可以未来的一些工作可以考虑以下的一些方案。

1）对于第一个问题来说，我认为可以对读和写做一个平衡，将 Wu 等人提出的方法与设置优先级的方法相结合。大部分时候当读取请求来临时，挂起正在进行的 P/E 操作，但是当某个写请求的优先级别很高时，则不暂停 P/E 操作，让读请求等待，从而不影响一些重要的写操作的性能。

2）随着存储技术的不断发展，未来的 SSD 制造工艺越来越成熟，存储硬件的成本会变低，这时通道的浪费带来的成本损失也会相应的降低，所以为了解决写操作时的尾延迟问题，可以减小 N 的大小，以额外的成本开销换取性能的提升。

3）同 2）理，我认为可以增大作为持久写入的缓存 RAM 等容量来解决频繁写操作造成的延迟的问题，虽然这样的方法不能从根本上克服这个缺陷，但也是一种可行的方案。

4）对于跳过 ECC 检查来说，如果没有 ECC 检查，就会影响数据的完整性，但是由于 Yan 等人采取的策略依赖于平面内回拷的操作，不经过控制器，所以无法进行 ECC 检查。针对这样的问题，我认为可以在 SSD 周期性空闲时间内清理数据，强制所有的 Flash 页面流过 ECC。这样就能尽量降低数据出错的频率。这种做法对性能的影响还需要将来的研究来考证。

参考文献

[1] Yan S, Li H, Hao M, et al. Tiny-Tail Flash: Near-Perfect Elimination of Garbage Collection Tail Latencies in NAND SSDs[J]. Acm Transactions on Storage, 2017, 13(3):1-26.

[2] Guanying Wu and Xubin He. Reducing SSD Read Latency via NAND Flash Program and Erase Suspension. In Proceedings of the 10th USENIX Sympo sium on File and Storage Technologies (FAST),2012.

[3] Gala Yadgar, Eitan Yaakobi, and Assaf Schuster. Write Once, Get 50% Free: Saving SSD Erase Costs Using WOM Codes. In Proceedings of the 13th USENIX Symposium on File and Storage Technologies (FAST), 2015.

[4] ARASE, K. Semiconductor NAND Type Flash Memory with Incremental Ste p Pulse Programming, Sept. 22 1998. U.S. Patent 5,812,457.

[5] KIM, Y., ORAL, S., SHIPMAN, G., LEE, J., DILLOW, D., AND WANG, F. Harmonia: A Globally Coordinated Garbage Collector for Arrays of Solid-State Drives. In MSST (2011), IEEE, pp. 1–12.

[6] ONFI WORKING GROUP. The Open NAND Flash Interface,2011. http://onfi.org/.

[7] Mingzhe Hao, Gokul Soundararajan, Deepak Kenchammana-Hosekote, Andrew A. Chien, and Haryadi S. Gunawi. The Tail at Store: A Revelation from Millions of Hours of Disk and SSD Deployments.In Proceedings of the 14th USENIX Symposium on File and Storage Technologies (FAST), 2016.

[8] aeho Kim, Jongmin Lee, Jongmoo Choi, Donghee Lee,and Sam H. Noh. Improving SSD Reliability with RAID via Elastic Striping and Anywhere Parity. In Proceedings of the International Conference on Dependable Systems and Networks (DSN), 2013.

[9] QURESHI, M., AND ET AL. Improving Read Performance of Phase Change Memories via Write Cancellation and Write Pausing.In HPCA (2010), IEEE, pp. 1–11.

[10] AGRAWAL, N., PRABHAKARAN, V., AND ET AL. Design Tradeoffs for SSD Performance. In USENIX ATC (Boston, Massachusetts,USA, 2008).

[11] KIM, Y., ORAL, S., SHIPMAN, G., LEE, J., DILLOW, D., AND WANG, F.

Harmonia: A Globally Coordinated Garbage Collector for Arrays of Solid-State Drives. In MSST (2011), IEEE, pp. 1–12.

[12] Report: SSD market doubles, optical drive shipment rapidly down. http://www.myce.com/news/reportssd-market-doubles-optical-drive-shipmentrapidly-down-70415/, 2014.

[13] BREWER, J., AND GILL, M. Nonvolatile Memory Technologies with Emphasis on Flash. IEEE Whiley-Interscience, Berlin(2007).

[14] WU, G., HE, X., XIE, N., AND ZHANG, T. DiffECC: Improving SSD Read Performance Using Differentiated Error Correction Coding Schemes. MASCOTS (2010), 57–66.

致谢

结课论文到此就要结束了，从几个月之前开始初步接触存储领域，到现在报告即将完成，这期间的学习过程都还历历在目，经过课堂的学习和课后论文的阅读，总算大致了解了存储领域的小块知识。

感谢这两个多月来何水兵老师的悉心指导，何水兵老师治学严谨，专业知识过硬，上课讲的内容紧跟时代潮流，给我们介绍了很多存储领域最新的学术研究动态，与此同时，也不忘给我们讲授一些基本的知识，给我们打了一个好的基础，有了这些基础才能使我们这些原本的门外汉能够去阅读一些相关的论文。除此之外，老师授课方式不是一层不变，通过后来的小组展示环节，大家互相分享，交流意见，拓宽视野，学到了除了自己研究方向之外的知识，我从中得到了很大的收获。

最后，我要感谢所有在课堂上帮助过我的同学们和老师，再一次对你们表示衷心地感谢！