智能手机应用程序的I/O特性及其对eMMC设计的影响

邓州
圣地亚哥州立大学
电子邮件： zhoud@rohan.sdsu.edu

温潘、王伟
圣地亚哥州立大学
电子邮件： {pan，王}@rohan。sdsu.edu

谢涛
圣地亚哥州立大学
电子邮件： txie@mail.sdsu.edu

摘要：绝大多数智能手机都使用eMMC（嵌入式多媒体卡）设备作为其存储子系统。最近的研究表明，存储子系统是影响智能手机应用程序性能的一个重要贡献者。然而，对于智能手机应用程序的块级I/O特性及其对eMMC设计的影响仍然知之甚少。在这项研究中，我们收集并分析了Nexus 5智能手机上的18个常见应用程序（如电子邮件和推特）的块级I/O痕迹。我们观察到一些I/O特性，从中推导出了eMMC设计的几个含义。例如，我们发现，在18个跟踪请求中，有15个大多数请求（44.9%- 57.4%）都是小的单页（4KB）请求。这意味着应该快速服务小请求，从而提高eMMC设备的整体性能。接下来，我们将进行一个案例研究，以演示如何应用这些含义来优化eMMC设计。受到两个含义的启发，我们提出了一个混合页面大小（HPS）的eMMC。实验结果表明，HPS方案可以平均减少86%的响应时间，同时提高24.2%的空间利用率。

Ⅰ.介绍

智能手机的存储子系统通常通过闪存文件系统[1]或像Ext4 [2]这样的块文件系统使用NAND闪存。早期基于android的智能手机广泛使用YAFFS2作为默认文件系统，这是一个专门的文件系统，用于存储原始闪存。在安卓2发布后。3（姜饼）平台在2010年，当基础存储子系统从原始闪存更改为eMMC设备时，基于安卓系统的智能手机的默认文件系统被切换到Ext4 [3]。eMMC具有一个统一的协议，并为Ext4提供了一个传统的块设备接口。它的控制器在本地处理地址映射、磨损均衡和垃圾收集，这在很大程度上减轻了上述文件系统的负担。图1显示了像Nexus 5这样的智能手机的I/O堆栈。大多数智能手机应用程序的文件和数据都由SQLite库管理，这是Androidbase智能手机的默认数据库管理系统（DBMS）。通常，一个应用程序的一个I/O活动会导致由内核层上的虚拟文件系统（VFS）所服务的多个SQLite I/O请求。每个SQLite I/O请求通常调用一个由像Ext4这样的块文件系统提供的系统调用函数，该函数进一步生成I/O请求并将它们发送到块层。块层负责调度这些I/O请求，这些请求最终被分配到一个eMMC驱动程序。eMMC驱动程序将每个I/O请求转换为由eMMC控制器所理解的命令，该控制器执行所有命令以满足I/O请求。

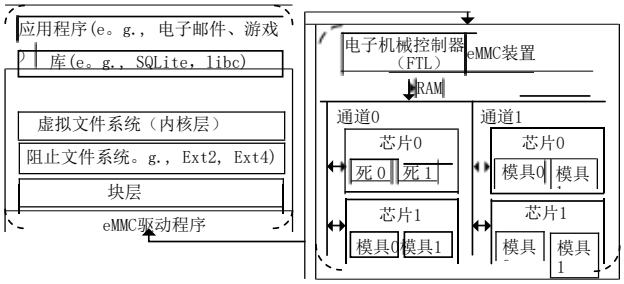

图1：安卓I/O堆栈。

类似于基于闪存的固态驱动器（SSD），eMMC设备的行为类似于传统的块设备
它自己的flash转换层（FTL）隐藏了所有的flash约束，比如在写前擦除。由于智能手机在空间、功率和成本上的限制，与SSD相比，eMMC设备具有更简单的FTL和架构，以及更小的RAM缓冲区。因此，eMMC设备的性能远低于SSD [4]。另一方面，近年来，智能手机的软件和硬件技术发展迅速。一些移动操作系统，如谷歌安卓系统、苹果iOS系统和微软的移动操作系统已经被开发出来，以改善硬件管理和对应用程序的支持。在硬件方面，嵌入式多核处理器的性能可以与个人计算机CPU相媲美，[5][6]引入了[5][6]级的RAM。这些新进展使eMMC设备成为当代智能手机[7][8]的性能瓶颈。显然，eMMC的性能还需要进一步提高。

为了开发一个高性能的eMMC设备，更好地了解智能手机应用程序的块级I/O特性是必不可少的。虽然最近有文献报道了一些关于智能手机存储子系统的研究，但没有一篇关注如何基于智能手机应用的I/O特性优化存储子系统设计。据我们所知，唯一与这项研究相接近的工作是[10]，它分析了Nexus S智能手机上的14个安卓应用程序的I/O行为。它的主要发现是，SQLite和Ext4的组合操作对基于NAND的存储产生了不必要的过度写操作，这不仅降低了I/O性能，而且显著降低了底层NAND闪存的生命周期

$31.00©2015 IEEE
DOI 10.1109/IISWC.2015.8

存储[10]。因此，它建议SQLite和Ext4需要以一种集成的方式进行优化，以消除冗余的工作。在本研究中，我们的目标是利用块级I/O特性的含义来优化eMMC设计，而不是关注理解软件层[10]之间的交互。

为了实现这一目标，我们首先实现了一个I/O监控软件工具，它使我们能够从18个应用程序中收集25个块级I/O跟踪。e.,在Nexus 5智能手机上的18个单独的痕迹)和它们的组合（即7个组合痕迹）。接下来，我们定量地分析了这些痕迹，并发现了一些有趣的智能手机应用程序的I/O特性。例如，我们发现，在18个单独的跟踪中，有15个单独的大多数请求（44.9%-57.4%）都是小的单页（4KB）请求。此外，在大多数跟踪中，超过80%的请求在到达后可以立即送达。此外，基于我们发现的I/O跟踪特性，我们得出了eMMC设计的几个重要意义。例如，发现大多数请求都是小请求的一个含义是，应该快速地服务这些小请求，从而可以提高eMMC设备的整体性能。最后，我们进行了一个案例研究来演示如何应用这些含义来优化eMMC设计。受到两个含义的启发，我们提出了一个混合页面大小（HPS）的eMMC方案。实验结果表明，与传统的纯4kb页面大小的结构相比，HPS方案可以将平均响应时间减少高达86%。与现有的纯8kb页面大小的架构相比，HPS可以提高高达24.2%的空间利用率。

本文的其余部分组织如下。在下一节中，我们将介绍跟踪收集的详细信息。第三节分析了25个轨迹的特征。第四节提供了这些特征的含义。第五节介绍了一个案例研究，以说明如何利用这些含义来优化eMMC设计。
我们在第六节中简要地总结了相关的工作。最后，我们在第七节中总结了本研究。

II .微量收集

在本节中，我们首先介绍跟踪收集的环境设置。接下来，我们介绍了一个名为BIOtracer（块级I/O示踪器）的I/O监视器的设计和实现，它收集了25个块级I/O跟踪。最后，我们将解释如何以单独和组合的方式使用这18个常见的应用程序，以便记录它们的块级I/O活动。

A.环境设置

我们使用Nexus 5作为我们的移动平台来收集所有的痕迹。它的存储子系统是一个32 GB的SanDisk INAND eMMC 4.51，没有外部sd卡。这款智能手机配有安卓v4.4版（KitKat）。在Nexus 5上，我们安装了定制的内核，并且我们使用了默认的配置。在每个跟踪收集过程中，一个I/O记录缓冲区被创建来记录I/O活动，其大小被设置为32 KB，它可以存储大约300条请求记录。在每个跟踪收集过程的开始时都会生成一个日志文件，以便可以定期将存储在I/O记录缓冲区中的I/O记录刷新到其中。对于每个跟踪收集过程，系统都将重新启动

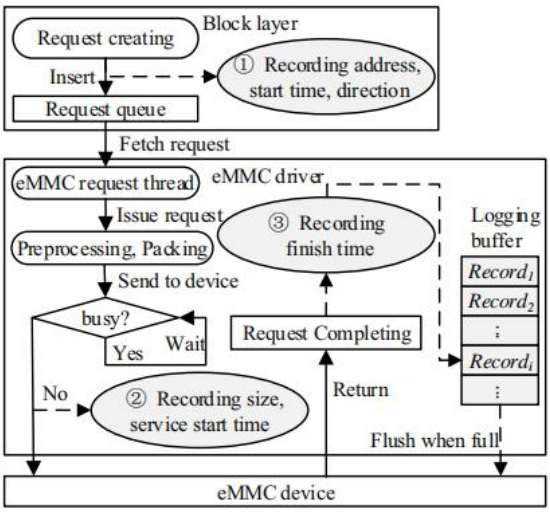

图2：生物生物示踪剂的工作流程。

并清除日志文件，以尽量减少来自其他进程的干扰。

B.BIO示踪剂

我们在Linux内核3.4.0中实现了一个名为BIOtrracer的跟踪收集器工具，以从这两者中收集I/O信息
块层和eMMC驱动层(见图。1).其体系结构如图所示。2.对于每个请求，它在块层上记录其开始时间、逻辑地址、请求大小和访问类型（读或写）(见图中的步骤1。2).在块层上创建请求后，将将其插入到块层中的请求队列中。然后，eMMC请求线程获取该请求，然后在进行一系列状态检查后将其发送给预处理功能。预处理函数将块请求转换为eMMC请求。如果可能，打包函数将将多个写入请求合并为一个大的写入请求。

最后，如果不忙，打包请求将发送到eMMC设备。否则，eMMC请求线程必须等待设备处于空闲状态。BIo跟踪器记录了请求确实发送到设备时请求的服务开始时间(见图中的步骤2。2).当设备驱动程序完成请求时，bio示踪器记录其完成时间(见图中的步骤3。2).请求的时间戳被合并到一个记录中，该记录被存储在主机的RAM中的一个32 KB的I/O记录缓冲区中(见图。2).当缓冲区满时，所有记录都被刷新到eMMC设备上的日志文件中。

C.BIOtracer开销分析

在本节中，我们将分析I/O监视器的开销。输入/O监控器引入了两部分的开销。第一部分是它的代码执行时间，它包括记录时间戳的时间和将它们存储到主存中的日志记录缓冲区所花费的时间。时间戳记录操作在三个不同的层中进行，如图所示。2.然而，在每一层中，I/O监视器只执行几行代码，而没有外设访问，

表一：选定的应用程序

应用程序	定义
无意义的	智能手机处于空闲状态
调用	接听来电
叫出	打个电话
引导	智能手机引导过程
电影	在智能手机上看电影
音乐	在智能手机上听歌曲
愤怒的小鸟	玩安格丽鸟游戏
摄像机视频	录制视频剪辑
谷歌地图	路线图和导航
消息传递	接收/发送、查看消息
推特	阅读和发布推文
电子邮件	接收/发送/查看电子邮件
脸书	查看图片/添加评论等。
亚马逊河	移动式网上购物
YouTube	在YouTube上观看视频
无线电广播	收听在线广播
安装	从谷歌播放中安装应用程序
Web浏览	在时代》网站上阅读新闻

等待和线程切换。因此，这部分CPU时间可以被安全地忽略，因为与内核中用于请求处理的数千行代码相比，用于时间戳记录的代码数量非常简单。

第二部分是日志记录缓冲区的冲洗成本。为了获得稳定的I/O模式，对于每个跟踪，I/O监视器会收集相对较长时间的I/O数据，包括应用程序的启动、运行和关闭。因此，I/O监视器会周期性地将缓冲区刷新到eMMC设备上。在我们的跟踪收集实验中，缓冲区大小被设置为32 KB，这可以容纳大约300个请求的记录。当缓冲区已满时，缓冲区中的所有数据都会存储到eMMC设备中。我们观察到，刷新操作总是会生成5-7个额外的I/O操作（例如，同步打开、附加和关闭日志文件）。因此，由I/O监视器引起的额外的I/O操作数平均约为6次，这就只有2% (i。e.,6/300 = 2%)的正常I/O请求的数量。

D.应用程序选择

在大量的智能手机应用程序中，我们选择了14个常用的应用程序。此外，我们还收集了4个系统功能的I/O活动，包括呼叫输入、呼叫输出、空闲和启动。为简单起见，这四个系统函数也被称为应用程序。表I总结了这18个应用程序，表II显示了如何使用这18个应用程序来生成这25个跟踪。Nexus 5智能手机的后台服务包括电子邮件和信息接收服务，以及一些基本的智能手机功能，如网络连接。除了CallIn和CallOut应用程序外，在应用程序运行跟踪收集时启用所有这些后台服务。

III .跟踪分析

在本节中，我们分析了25条轨迹，包括18个单独的轨迹和7个组合轨迹。而每个人
跟踪来自于一个特定的应用程序，一个组合跟踪(e。g.，音乐/WB)是由两个同时运行的应用程序生成的，如音乐和网络浏览。为了充分了解I/O模式，我们不仅研究请求大小、请求类型和请求位置，而且还调查请求响应

表二：跟踪收集的详细信息

空闲状态（晚上10点到早上6点）：空闲状态。
启动(30秒)：启动智能手机。
打电话，打电话（1个小时）：模拟一个电话采访，包括回答、交谈、倾听和闲逛。
谷歌地图（0.5 - 1小时）：录制视频，玩游戏，驾驶导航。
脸书、推特、亚马逊、电子邮件、信息传递（10 - 20分钟）：查看评论，搜索人或物品，查看图片，并撰写回复。
网络浏览，YouTube，广播，音乐（1 - 1.5小时）：阅读新闻，观看在线视频，听广播，听音乐。
电影，安装（10分钟）：观看本地存储的电影，通过WIFI连接安装游戏应用程序。
除FB/Msg（10分钟至半小时）外的组合跟踪：在听广播或音乐时使用Facebook、消息或浏览在线新闻
FB/短信（12分钟）：使用Facebook，每当有新的按摩出现时，就会切换到阅读信息，回复后继续使用Facebook。

时间、服务时间和到达间隔时间。此外，我们还分析了请求并行度的程度。

A. eMMC吞吐量

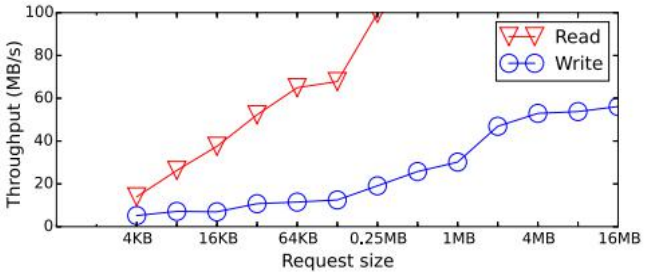

图3：请求大小对吞吐量的影响。

图3表明，请求大小在吞吐量方面显著影响了eMMC的性能。在此图中，通过计算在所有跟踪中具有该大小的请求的平均访问率来获得特定请求大小的吞吐量。在收集到的25条跟踪中，读取请求的最大大小是256 KB，而写请求的最大大小是16 MB。这就是为什么读取吞吐量曲线终止在大约100 MB/s(见图。3).图3显示读取吞吐量从13.94 MB/s变化到99.65 MB/s，而写吞吐量从5变化。18 MB/s至56。当请求大小从4 KB增加到16 MB时，MB/s为15 MB/s。当请求大小为256 KB时，读取吞吐量达到其最大值(i。e.,99.65 MB/s)，而写吞吐量只有19 MB/s。这是因为在闪存中阅读页面比写页面要快得多。我们还发现，更大大小的请求（例如，超过1 MB）会导致读和写的更高的吞吐量，这归因于eMMC驱动程序层上的打包命令。此外，图。3也证实了eMMC的性能明显低于SSD，后者在顺序读取和写时分别能达到550 MB/s和520 MB/s，[11]。

B.大小相关分析

收集到的25个轨迹的请求大小相关特征见表三。图4说明了18个单个跟踪的请求大小分布。在表三中，数据大小列和请求的数量。列存储

表三：这25条轨迹的与大小相关的统计数据

应用程序名称	数据尺寸（KB）	数量的要求。	最高的尺寸（KB）	大街尺寸（KB）	大街R尺寸（KB）39。5	大街W尺寸（KB）	写要求。百分数(%)	写尺寸百分数(%)
无意义的	123,220	6,932	1,536	17.5	12.0	15.0	88.94	75.41
呼叫呼叫启动电影音乐	27,300	1,491	1,536	18.0	10.0	18.0	99.93	99.96
奶奶的相机	27,364	1,569	1,536	17.0	61.0	17.5	98.92	99.37
视频谷歌地图消息推特	982,200	18,417	1,536	53.0	27.5	37.5	33.07 5	23.26 3
电子邮件脸书亚马逊	130,420	4,781	20,816 512	27.0	62.5	17.0	52.80	14.48
YouTube广播	240,060	6,913	940	34.5	51.0	.59	84.51	73..3712
安装网络浏览音乐/WB广播/WB广播/FB广播/FB音乐/味精广播/味精FB/味精	94,684	3,215	3,940	29.0	38.5	25.0	29.46	88.85
	2,283,184	9,348	10,104	244.0	28.5	736.5	86.78	75.90
	197,808	12,603	8,174	15.5	23.0	13.5	97.30	94.38
	63,668	5,702	128	11.0	35.5	10.5	88.48	69.86
	187,540	13,807	2,216	13.5	14.5	10.5	70.37	78.62
	59,276	2,906	388	20.0	28.5	22.5	74.42	70.70
	97,436	3,897	2,680	25.0	24.5	23.5	63.02	55.07
	67,412	3,272	1,392	20.5	19.5	18.0	97.50	96.46
	28,692	2,080	1,536	13.5	36.0	13.5	98.68	97.59
	115,972	5,820	11,164	19.5	22.0	19.5	98.26	99.58
	1,653,900	17,952	22,144	92.0	21.5	93.0	80.71	81.95
	95,908	4,090	1,536	23.0	50.5	23.5	81.68	57.36
	289,280	12,603	1,56 44	21.5	29.0	15.0	72.02	63.65
	269,932	5,702	2,716	22.5	38.0	19.5	87.67	62.34
	442,388	13,807	2,424	12.5	23.0	8.5	91.68	86.92
	153,776	2,906	1,368 472	14.5	56.0	13.5	94.43	77.96
	234,000	3,897	1,536 732	14.0	17.5	11.5	98..4015	97.55
	150,344	3,272		13.5	21.5	13.0	84.72	71.72
	182,632	2,080		11.5		9.5		

分别访问数据的总大小和请求总数。“最大大小”列记录在跟踪中找到的最大请求大小。*写入请求。百分数*列记录每个跟踪中写请求的百分比，而写入大小百分比。显示写入数据量占访问数据总大小的百分比。

由于打包命令，大多数跟踪中的最大请求都大于512 KB，这是Linux内核中允许的最大请求大小。此外，Ave。大小字段显示了数据库范围最大的应用程序(e。g.，它的平均请求大小比其他的要大得多。在18个单独的应用程序中，大多数的平均请求大小从11 KB到34.5 KB。

表三中最后两列中的值表明
除了引导、电影和音乐之外，大多数的痕迹都是写作主导的。在启动过程中，会发出许多读取请求来加载程序和配置文件，以便初始化智能手机。音乐和电影还需要从eMMC设备获取媒体数据，从而产生大量的读取操作。CallIn和CallOut产生的读取操作很少（分别小于1%和2%）。这是因为，当用户调用某人时，除了一些日志文件更新外，所有其他活动都将处于待定状态。AngryBrid不断更新其日志和播放状态，从而产生大量的写操作。此外，
相机视频有88.85%的数据
请求百分比只有29.46%，因为它主要发出大的顺序请求，这可能会在驱动程序级别进一步打包。电子邮件等在线应用程序主要是从互联网上获取数据。为了更新日志并将数据缓存到eMMC设备，这些应用程序通常会发出大量的写操作。

特点1：大多数智能手机应用程序都占主导地位。在18个单独的跟踪中，有15个中的写请求的百分比从52.8%到99.9%不等，其中
其中6个比例超过了90%。

18个应用程序的请求大小分布如图所示。4.请求被分为不同的范围(e。g., 小于或等于4 KB)，基于它们的大小。虽然一个范围代表一个连续的区域，但可能的大小只能是4的倍数，因为所有的请求大小都与flash页面大小对齐(i。e.,在文件系统级别。图4显示，除了电影和引导之外，在18个单独的跟踪中，有16个的小请求占总请求的大部分。尽管如此，像引导和电影这样的应用程序还是会生成大量的大型请求。总体趋势是，小请求占多数，而大请求的数量通常很少。与互联网有关的应用程序(i。e.,图中的最后10个应用程序。4)的分布也很相似，一般都遵循相同的趋势。我们发现，数据密集型跟踪具有其独特的请求大小分布，而其他跟踪共享类似的模式。例如，Movie有大量大小在16 KB到64 KB之间的请求。

特征2：在大多数应用程序中，小规模的请求占请求总数的很大比例。在18个单独的跟踪中，有15个，大多数请求（44.9% - 57.4%）是小请求（即4 KB）。

C.时间相关分析

在本节中，18种应用程序总结了与时间相关的I/O特性。5和无花果。6.*在表四中，从每个跟踪的开始时间到结束时间，测量记录持续时间字段中的值*。启动的持续时间由系统性能决定，而其他应用程序的持续时间则由用户决定。尽管应用程序的I/O模式会在很大程度上受到用户习惯的影响，但我们确保每个应用程序的持续时间足够长，从而可以记录一个稳定的I/O模式。

当到达率取为每秒到达的请求数，而数据访问率则定义为

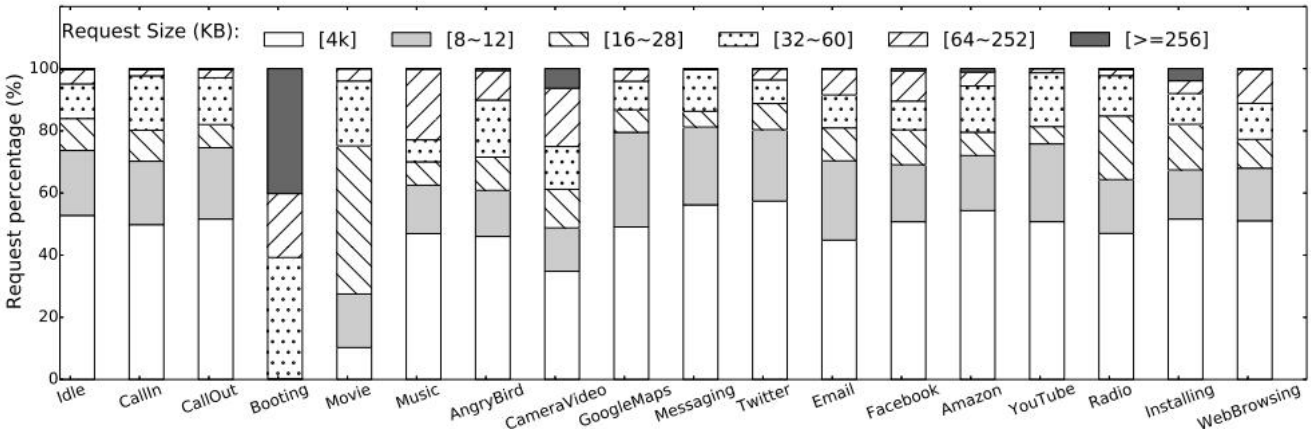

图4：请求大小分布。

表四：25条轨迹的时间相关统计数据

应用程序名称	录音持续时间第二	到达速率 (Reqs /s)	访问权限速率 (KB/s)	没有等待请求比率	平均值。保存 (ms)	平均值。回复。 (ms)	空间地点 (%)	时间的地点 (%)
无意义的	29,363	0.24	4.20	89	7.42	9.24	25.32	34.22
调用	3,767	0.40	7.25	98	5.61	6.18	29.59	31.00
叫出	3,700	0 42	7 40	94	5 57	6 07	27 29	35 14
引导	40	460.40	24,555.00	58	1.65	4.93	28.19	19.70
电影	998	4.79	130.68	23	2.13	6.28	17.25	1.72
音乐	3,801	1.82	63.16	64	2.38	3.45	21.51	31.86
安格里布里德	2,023	1.59	46.80	84	3.44	4.06	30.08	26.07
摄像机视频	3,417	2.74	668.18	47	8.07	11.61	20.34	16.30
谷歌地图	1,720	7.33	117.76	85	1.40	2.23	21.10	42.78
消息传递	589	9 68	108 10	86	1 68	1 88	28 85	50 82
推特	856	16.13	219.09	84	1.72	2.07	26.57	52.90
电子邮件	740	3.93	80.10	63	3.01	4.09	14.49	34.87
脸书	1,112	3.50	87.62	69	2.99	4.08	19.89	34.21
亚马逊河	819	3 90	84 29	73	1 45	4 70	17 79	26 38
YouTube	4,690	0.44	6.12	96	6.90	7.19	47.61	16.35
无线电广播	4,454	1.31	26.04	82	3.54	6.62	23.90	29.18
安装	977	18 37	1,692 84	80	3 64	10 04	22 59	49 57
Web浏览	4,901	0.83	19.57	79	4.33	5.20	23.77	30.83
音乐/WB	2,165	6.10	133.62	65	1.70	3.61	18.40	38.40
无线电/WB	1,227	9.78	219.99	69	1.86	3.30	18.66	28.48
音乐/FB	2,026	17 34	218 36	70	1 13	2 09	14 19	60 50
无线电/FB	900	11.66	170.86	78	1.64	2.58	19.12	52.70
音乐/新闻	926	17.82	252.70	74	1.36	2.19	20.68	53.84
电台/新闻	660	16 82	227 79	89	1 63	2 04	27 25	49 48
FB/Msg	699	22.32	261.28	72	1.23	1.90	15.80	54.04

平均被访问的数据量(i。e., 每秒的读写。CallIn、Idle和YouTube显示出非常低的请求到达率（例如，每秒少于一个请求）和数据访问率（例如，小于10 KB/s）。启动只能持续40秒。不过，它还是会生成大量的读取请求。它具有最高的请求访问率和数据访问率。安装跟踪还显示了较高的请求到达率和数据访问率，因为安装过程会导致软件下载和安装。再加上表三中的平均大小，我们可以发现具有更高的数据访问率(e。g., 引导、摄像和安装)也有更大的请求大小。更进一步，
我们发现18条个体痕迹中有15条到达
每秒的速率少于10个请求。*NoWait要求*。比率表示在它们到达时不需要等待的请求的百分比。这意味着这些请求可以立即送达，因为没有正在送达的请求。表IV显示了至少63%的请求

在18个个体痕迹中有15个属于这一类别，在18个个体痕迹中有10个有超过80%
可以立即提供服务的请求。

特点3：大多数请求一旦到达，就可以立即得到服务。换句话说，同时到达的请求很少。

平均服务程序中的数据。(i.e., 平均服务时间)和平均Resp。(i.e., 表四中的平均响应时间)表明，来自启动、电影、Amazon和安装的请求具有更长的请求队列，因为它们的平均响应时间与平均服务时间的比例更高。结合Amazon不是一个数据密集型应用程序（见表三），我们可以得出Amazon具有不同的I/O模式。此外，我们注意到空闲、呼叫、呼叫、YouTube和网络浏览的请求到达率低于每秒1个请求。它们也有更高的平均响应时间和平均服务时间

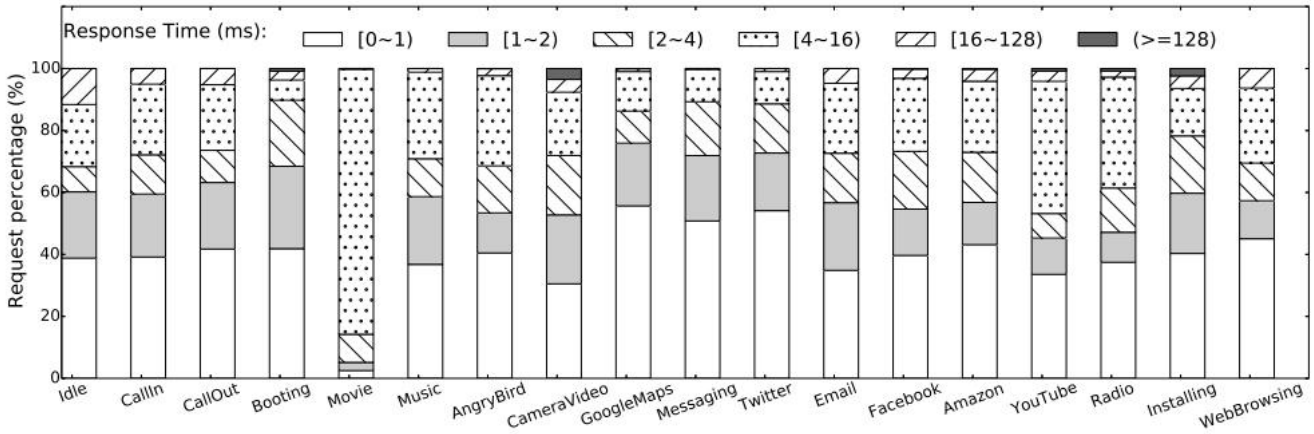

Fig. 5: Request response time distributions.

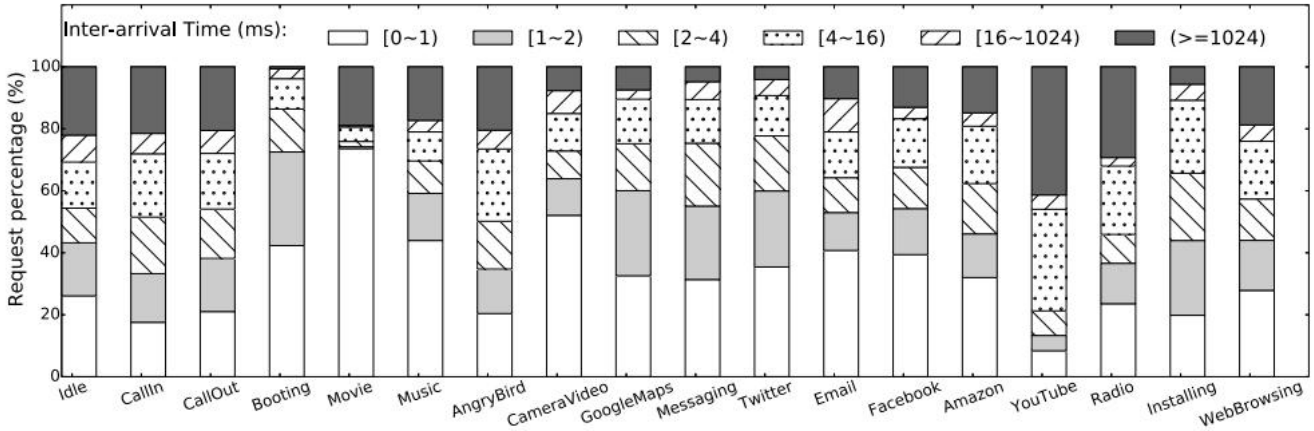

图6：请求到达时间分配。

其他（e。g.，音乐、电子邮件、脸书)。这是因为，当一段时间内没有请求到达时，eMMC设备将进入低功耗模式。因此，eMMC设备需要一个相对较长的预热时间来满足新到达的请求。在大多数其他跟踪中，平均响应时间的值大约是平均服务时间的两倍。这一事实加上高的等待要求。比值值意味着花费在软件层的时间是显著的，因此需要进一步优化。

特征4：一个eMMC设备将进入一个较低的-
如果请求间到达时间超过其省电阈值，则为电源模式。因此，在某些应用程序中，可能会发生周期模式切换。然而，频繁的模式切换，增加了请求的平均响应时间。

最后两栏记录了这18个国家的位置
单个的痕迹。空间局部性定义为顺序请求访问占跟踪中请求总数的百分比。当当前请求的起始地址位于其前身的结束地址旁边时，就会发生顺序请求访问。时间局部性是地址命中数占请求总数的百分比。

当重新访问一个地址时，地址命中的次数会增加1。在空间局部性方面，18条个体轨迹中有16条的空间局部性低于30%，18条轨迹的空间位置均低于48%。与空间局部性相比，时间局部性总体上略好（例如，18条轨迹中有11条的时间局部性在30.83%到52.9%之间）。然而，与在一些服务器类应用程序中观察到的20/80的位置规则相比，所有这些位置都相对较低。

特征5：这18个应用程序的位置一般较弱。此外，空间位置也低于时间位置。

图5提供了一些从表4中无法观察到的关于响应时间的信息。图中的总体趋势。5是大多数请求可以在2 ms内完成。此外，绝大多数请求可以在16个ms内处理。很少有长时间的反应时间。e.，超过128个ms)的请求存在于18个跟踪中。我们发现响应时间分布与请求大小分布密切相关。高相关性表明，一个请求的响应时间在很大程度上取决于它的大小

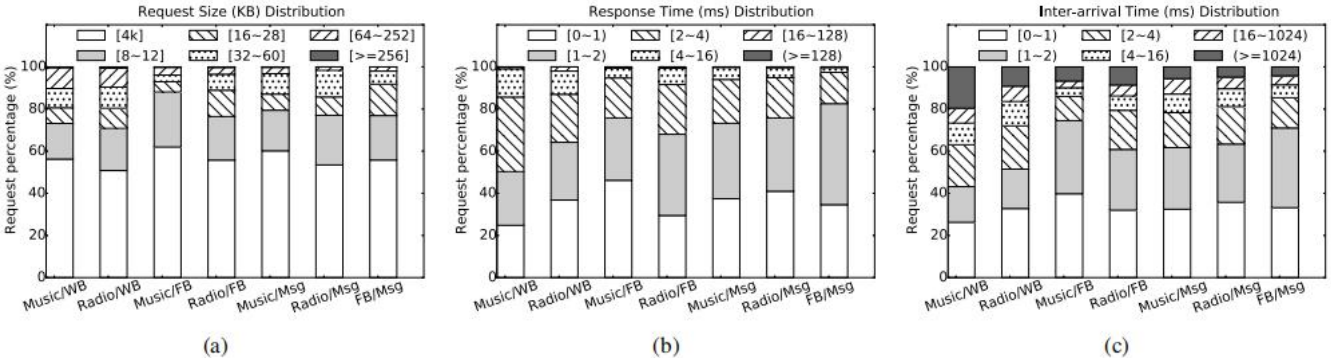

图7：7个组合轨迹的I/O模式。

进一步意味着在请求队列中等待的请求很少。例如，Movie显示了大小在16到64 KB之间的大量请求（即超过65%）。它在从4 ms到8 ms的范围内有近70%的响应时间。

图6提供了到达间时间分布的详细信息。呼叫输入和呼叫输出的大量到达时间表明，在电话呼叫过程中的I/O活动较少。网络浏览、YouTube和Radio也表明，他们没有严重强调eMMC设备，因为他们有更大的间隔到达时间。此外，在观看电影时，大多数到达时间都小于1 ms(见图。6).这一数字进一步支持了与互联网相关的应用程序具有相似的I/O模式的说法，因为它们显示了相似的到达时间间隔分布。与YouTube和Radio等在线应用程序相比，本地应用程序（如启动、电影、音乐、相机视频）显示出更小的相互到达时间。

特征6：在大多数应用程序中，平均请求间到达时间较长。18个应用程序中有13个应用程序的平均请求间到达时间至少为200 ms。在18条痕迹中有10条，超过20%的到达时间大于16个ms。

D.并发应用程序的输入/输出模式

在大多数情况下，智能手机用户通常会运行一个应用程序(e。g., YouTube)，一次启用了后台服务。不过，用户同时启动多个应用程序并不少见。例如，用户可以在听音乐（即音乐）时在网站上阅读新闻（即网络浏览）。另一个例子是，当用户使用Facebook（即Facebook）时，他可能会突然切换到阅读传入的信息(i。e., 消息传递）。然后，在他回复了这个信息后，他就可以继续使用脸书了。因此，除了从18个应用程序中的一个中收集18个单独的跟踪外，我们还从7个常见的应用程序组合中收集了7个组合跟踪，以了解它们的I/O模式。这7个应用程序组合包括{音乐、广播}×（FB）、消息（Msg）、网络浏览（WB）}以及脸书和消息之间的任务切换。e., FB/Msg)。之所以选择音乐和电台，是因为它们是其他应用程序的热门伙伴，比如网络浏览。图7显示了7个组合的特点

根据请求大小和时间进行跟踪。图7a说明，与包含无线电的组合跟踪相比，包含音乐的组合跟踪具有更高的4 KB大小请求百分比。然而，7个组合跟踪的总体请求大小分布与15个单独的跟踪相似(见图。4).另一方面，图。7b表示，与单个应用案例相比，组合跟踪的响应时间并没有明显增加(见图。5).例如，音乐/WB的平均响应时间为3.61 ms，而Music本身的平均响应时间为3.45 ms，而网络浏览的平均响应时间为5.2 ms。此外，7个组合轨迹的到达时间一般较大，除Music/FB外，它们的到达时间都超过20%(见图。7c).

7个组合轨迹的平均到达时间范围从44.8 ms到164 ms。在比较请求到达率和数据访问率下的数据（见表四列3和4）后，可以发现，由于内存缓冲区等共享资源有限，组合跟踪在这两个项目中的值比两个单独跟踪的和的值更高。例如，虽然音乐/FB的访问率为218.36 KB/s，但音乐的访问率之和(i。e.,63.16个KB/s)和脸书(i。e.,87.62 KB/s)，仅为150.78 KB/s。*然而，NoWaitReq的高值*。比率（见表IV列5）表明，对于并发应用程序，对于同时到达的多个请求仍然不需要并行处理机制。总之，来自应用程序组合的7个组合轨迹证实，即使当多个应用程序同时运行，智能手机应用程序的I/O特性通常都是稳定的。

增值对EMMC设计的影响

基于这6个特点，我们能够为eMMC设计师提供以下含义：

含义1：一旦超过80%的请求以大多数路径到达，它们就可以立即得到服务，这意味着只有很少的请求同时到达（特征3）。这意味着在设备级别上增强并行性(e。g., 使用外部SDcard)或在操作系统层提供并行请求队列并不能帮助提高性能。这是因为它在智能手机上的性能明显低于

内部eMMC设备。例如，我们观察到Nexus 5上eMMC的性能大约是主流制造商提供的8个[7]测试的最佳性能的三倍。对于大多数跟踪，使用外部SDcard可能会意外地降低整体性能，因为当内部eMMC设备能够及时处理大多数请求时，较慢的外部SDcard会对整体性能产生负面影响。减少大规模请求的服务时间可以显著提高整体性能。此外，特征3还意味着请求的响应时间主要由其服务时间决定。这一事实表明，大规模的请求有更长的响应时间，因为现有的eMMC设备通常有非常有限的通道数量。例如，来自SanDisk的最新eMMC产品只提供了两个通道的[12]。因此，从一个大规模请求中分离出来的多个子请求（例如，2多个子请求）不能以完全并行的方式处理。

含义2：eMMC中的FTL需要进行定制，以匹配智能手机的I/O特征（从特征3和特征6开始）。我们发现，18条跟踪中有13条的请求间到达的平均时间超过200毫秒，这足以让一个垃圾收集过程完成。因此，应该重新设计FTL中的垃圾收集机制，以便在执行这些非数据密集型应用程序时启动垃圾收集。这样一来，用户就无法感知到由于垃圾收集而导致的性能下降。在SSD FTL中，当空闲块的数量达到预定义的阈值时，通常会触发垃圾收集。eMMC的FTL不应该采用相同的策略，因为在空闲块的数量变得显著较低之前，有大量的机会进行垃圾收集。

含义3：我们观察到，几乎所有的轨迹的时间局部性和空间局部性都很弱（从特征5开始）。例如，大部分轨迹的空间位置小于30%，时间位置小于40%（见表4）。因此，由于命中率较低，在eMMC设备内的大尺寸RAM缓冲区可能并不有利于性能优化。

含义4：从特征5中，我们还注意到18条轨迹中有14条的时间局部性低于40%，16条轨迹的空间局部性低于30%（见表4）。低位置表明，在eMMC设备级别上的I/O请求倾向于访问闪存上的不同位置。因此，我们认为，一个简单的磨损水平调整策略是足够的
一个eMMC设备。

含义5：特征2揭示了，在18个跟踪请求中，有15个大多数请求（44.9%-57.4%）是小的单页（4 KB）请求。这意味着，快速服务于大量的小请求可以提高eMMC设备的整体性能。更好地服务于这些小请求的一种可行方法是使用SLC（单海拔单元）闪存，它比MLC flash具有更好的读写性能和更高的价格。幸运的是，MLC闪存单元可以通过有选择地使用其快速页面来在SLC模式下工作，从而获得类似于SLC的性能[13][14]。因此，可以以50%的容量损失为代价来实现性能的提高。

V.案例研究

在本节中，我们将进行一个案例研究，以演示如何利用这些含义所提供的见解来优化eMMC设备的设计。含义1表明，加快大规模请求服务时间可以显著提高eMMC设备的整体性能。显然，具有大页面大小的eMMC设备是可取的，因为它可以有效地处理大型请求。事实上，现代的ssd倾向于使用一个更大的flash页面大小的[15]。另一方面，含义5暗示，小型请求也应该迅速得到服务，因为它们是大多数智能手机应用程序中的主要请求。但是，具有大页面大小的eMMC设备不适合处理小页面大小的请求，因为它可能会降低eMMC设备的性能和使用寿命。例如，当跟踪中的大多数请求是随机的时，4 KB写入一个大页面大小的性能(e。g.eMMC可能低于小页面大小(e。g.,因为将一个4 KB的数据写入到一个
将8 KB页面比写到4 KB页面需要更长的时间。此外，当两个eMMC设备具有相同的总容量时(e。g., 与4kb-page大小的eMMC相比，8kb-page大小的eMMC的页面数量要少得多。因此，在它有限数量的空闲页面被小的随机写入请求快速消耗后，它将有更多的垃圾收集（GC）操作。更多的GC操作进一步降低了性能，并缩短了设备的使用寿命。因此，需要一个小页面大小的eMMC设备来处理小型请求。因此，含义1和含义5激励我们提出一个混合页面大小（HPS）eMMC，以有效地同时服务于小尺寸和大尺寸的请求。HPS的基本思想是，一个eMMC中的所有块都具有相同的页面数量（例如，在我们的实验中是1,024个），而一个块中的所有页面都具有相同的大小(e。g.,4 KB).然而，页面的大小可能会在不同的模具块中有所不同(见图。1).据我们所知，HPS eMMC设备并不存在。因此，仿真成为验证其有效性的唯一途径。

表五：三种eMMC设备的配置

	4PS	8PS	高压蒸汽
页面读取延迟（我们）	160	244	N/A
页面写入延迟（我们）	1,385	1,491	N/A
块擦除延迟（我们）	3,800	3,800	3,800
Channel×chip ×die ×plane	2×1 ×2×2	2×1 ×2×2	2×1 ×2×2
每个平面的块	1,024	512	错误+256 8kb页的错误
每块页数	1,024	1,024	1024
总容量	32 GB	32 GB	32 GB

A.仿真设置

由于其成本限制，一个eMMC设备通常不超过两个通道。每个通道上都附有一个或多个MLC闪存芯片。每个芯片由多个模具组成，每个模具有数千个块。每个区块都有数百页。由于eMMC设备可以被视为一个轻量级的SSD，所以我们使用一个名为SSDsim [16]的经过验证的SSD模拟器来模拟eMMC设备。SSDsim是一个事件驱动和高度精确的模拟器。我们在SSDsim中实现了HPS方案，这样它就可以在一个模具中支持不同的页面大小的块。我们配置了几种混合页面大小的芯片

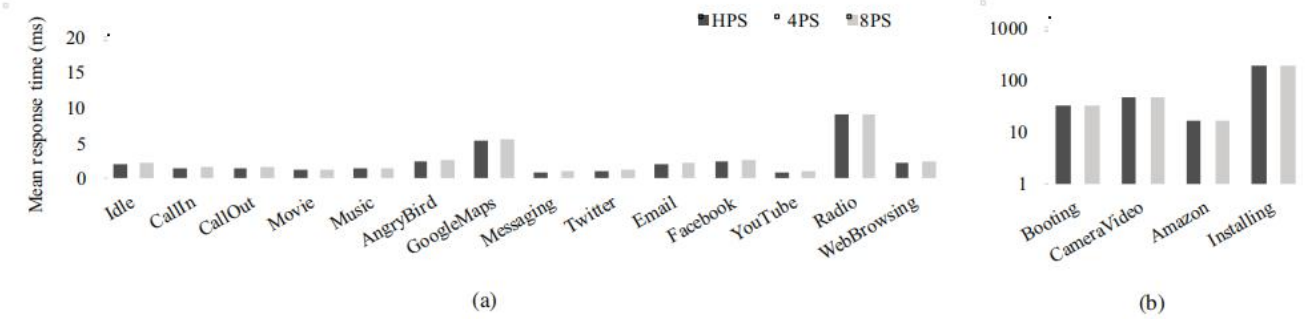

图8：三种方案之间的性能比较。

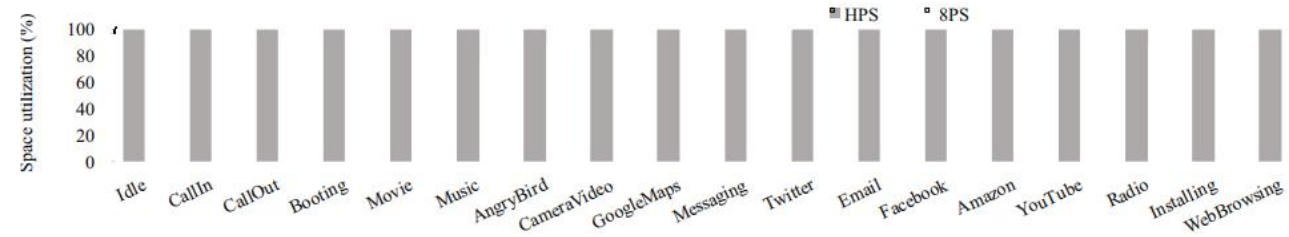

图9：8PS与HPS的空间利用率比较。

在eMMC设备中，开发了一个新的请求分发器，将读/写请求发送到适当的页面。在HPS方案中，使用了两种大小的页面(i。e.,4 KB和8 KB)。与延迟相关的参数从微米数据表[17][18]中获得。

对于HPS方案，我们假设页面大小在一个平面内不同的块之间有所不同。例如，在模拟器配置中，我们假设在一个平面内有512 4KB块和256 8KB块(见图。10).我们还配置了一个纯4 KB页面方案（4PS）和一个纯
8 KB页面方案（8PS）作为两个基线eMMC设备。HPS和两种纯方案的详细配置如表V所示。请注意，这三种方案都有相同数量的通道、芯片、骰子和平面，因此内部并行性将对三种方案的性能产生相同的影响。根据配置，三种方案的总容量也相同（见表V）。

请求分发器将一个请求分割为多个页面。对于读取请求，基于由eMMC控制器维护的映射表来检索数据。对于一个写请求，它将根据其大小以不同的方式进行处理。例如，当一个写入请求的大小为20 KB时，它将被分为两个8KB子请求和一个4KB子请求。在单页大小的eMMC设备上，一些闪存空间被浪费了。例如，如果我们使用一个8kb页面大小的芯片，那么需要3个子请求（总共24 KB）来完成20 KB的写请求。结果，浪费了4 KB的闪光空间。写请求的空间利用率定义为20/24，等于83.3%。跟踪的空间利用率定义为其写入的数据总量与消耗的闪存空间总量的比率。显然，空间利用率越高，就表明eMMC设备的使用寿命越长。

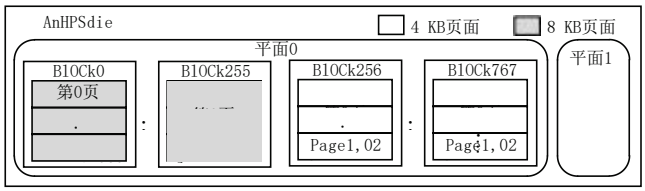

图10：一个HPS模具的结构。

B.实验结果

我们根据平均响应时间（MRT）和空间利用率来衡量每个方案的性能。所有的痕迹都会在一个模拟的全新eMMC设备上回放。模拟器的RAM缓冲层被禁用，以消除其对性能的影响。图8证明了HPS方案在所有18个跟踪中都优于4PS。由于启动、相机视频、亚马逊和安装的mrt远高于其他的，它们的结果以对数尺度表示，并在图中分离。8b.其余14个结果如图所示。8a.与4PS相比，HPS在启动MRT技术方面获得了最显著的改善（86%）。即使在最坏的情况下，电影跟踪，HPS仍然可以减少MRT24.0%。HPS平均可以实现61.9%的性能改进。8PS方案在MRT方面与HPS具有非常相似的性能。

图9显示了两种方案的空间利用率。我们使用4PS作为基线方案。HPS和8PS的结果均归一化为4PS的结果。由于HPS方案总是能实现与4PS相同的空间利用率，所以我们忽略了图中4PS的结果。9.与8PS相比，HPS

在音乐轨迹中，空间利用率提高最为显著，为24.2%。HPS方案平均达到13个目标。与8PS相比，空间利用率提高了1%。

VI .相关工作

随着智能手机越来越受欢迎，了解其存储子系统与系统性能之间的关系，最近开始引起业界和学术界的关注。截至2010年9月，由于安卓是一个开源操作系统[19]，基于安卓的智能手机的日销量为20万，而iPhone为8万。因此，几乎所有现有的关于智能手机存储子系统的工作都是在基于android的平台上进行的。然而，据我们所知，只有少数研究调查了基于android的存储子系统。金等。*al.*发现存储性能确实会影响一些常见应用程序的性能，如网页浏览、谷歌地图、应用程序安装、电子邮件和脸书[7]。他们观察到，仅仅通过改变底层的闪存存储，WiFi的性能在不同的应用程序中通常可以在100%到300%之间变化。Kim和Shin在研究了eMMC的内部特征后，证实了[7]的结论。特别是，他们研究了LSB备份、打包命令和弹性程序组在基于安卓系统的智能手机上的影响。他们得出的结论是，像eMMC设备这样的存储子系统需要进一步优化[4]。金和拉马钱德兰重新检查了智能手机的操作系统存储软件堆栈，以提高存储性能[8]。他们实现了一个名为Fjord的框架，该框架可以提供一个细粒度的控制机制来权衡性能的可靠性。[7][8]和[4]都没有关注于分析eMMC级I/O痕迹及其含义。

Lee和Won分析了来自6个不同类别[10]的14个安卓应用程序的I/O行为。他们专注于跨软件层的交互：应用程序、安卓操作系统、文件系统和底层存储设备[10]。他们发现，SQLite和Ext4的操作极大地加重了存储空间的负担，因为它们产生了不必要的过度写操作[9]。虽然他们也分析了一些I/O特性，如I/O的大小和随机性，以及它们对存储设备的影响，但他们的主要重点是理解由应用程序和操作系统层生成的I/O活动，然后，如何优化这些软件层。相反，我们专注于如何将从I/O特性中得出的含义应用于eMMC设计。

罗马数字 7结论

eMMC设备的性能显著地影响了智能手机应用程序[7][8]的性能。不幸的是，在定量分析块级智能手机I/O特性及其对eMMC设计的影响方面的研究很少。为了理解智能手机应用程序的I/O模式，我们实现了一个名为BIOtracer的I/O监视器工具，并将其集成到Nexus 5上的安卓内核3.4中。接下来，我们对25条轨迹进行了全面的分析。已观察到6个I/O特性。接下来，根据这些特征得出出eMMC设计的5个含义。最后，我们进行了一个案例研究来演示如何应用这些含义来优化eMMC设计。

确认

这项工作是由美国赞助的。美国国家科学基金资助项目：CNS1320738。

参考文献

[1] W.王，D。周和T。*谢，“一个嵌入式存储框架，将每个原始闪存设备抽象为mtd”，在第8届ACM交互系统和存储会议论文集*。ACM，2015年，p。7.

[2] A.Mathur。曹，S。巴塔查里亚。扩张器，A。Tomas和L。Vivier，“新的ext4文件系统：现状和未来计划”，发表在Linux研讨会论文集，第1卷。2.西特塞尔，2007年，页。21－33.

[3] T.H.打开后，“安卓2.3姜饼使用ext4文件系统”，2014年，访问日期： 2015-426年。在线可用： http://www。h-online.com/open/news/item/ Android-2-3-Gingerbread-to-use-Ext4filesystem1152775.html

[4] H.金和D。申文，《优化安卓智能手机的存储性能》，第七次国际会议论文集。*关于普遍存在的信息管理与沟通*。ACM，2013年，p。95.

[5] Kenlo，“智能手机和平板电脑上的游戏：你需要知道的一切”，2014年，访问日期： 2015-4-26。在线可用的：http://社区。吉夫格夫。博客/RAM-On-Smartphones-amp-Tablets-Everything-You-Need-To-Know/ba-p/7950298

[6]语音彩蛋，“最快处理器”，2014年，访问日期： 2015-4-26日。在线可用：http://我们。phonegg.com/top/53-最快处理器

[7] H.金，N。阿格拉瓦尔和C。*《为智能手机恢复存储》，《ACM存储交易》，第1卷*。8、没有。4，p.14，2012.

[8] H.金和美国。拉马钱德兰，“峡湾：智能手机的知情存储管理”，发表在大规模存储系统和技术（MSST），2013年IEEE第29届研讨会。IEEE，2013，页。1－5.

[9] S.JeongK。李，S。李，S。儿子和Y。获胜，“智能手机的I/o堆栈优化”，作为2013年USENIX年度技术会议的一部分提出。USENIX，2013年，页。309－320.

[10] K.李和Y。赢得，“智能层和愚蠢的结果：基于机器人的智能手机的Io特性”。*嵌入式软件*。ACM，2012年，页。23－32.

[11]三星，“三星850prossd”，2014年，访问日期： 2015-4-26.在线可用性： http://www.samsung.com/global/business/semiconductor/minisite/SSD/us/html/ssd850pro/overview.html？gclid= CjwKEAjwmfKpBRC8tb3Mh5rs23ASJACWylQPGj8I7uQtfTq_hMohfl7GF7ecW1b5usExP9lUZ1NtfxoCHk3w_wcB

[12] SanDisk，“2014年桑迪斯内部和极端”，2014年，访问日期： 2015-4-26。在线可用性： http://www.anandtech.com/show/7790/sandisk-inand-extreme-in-2014finally-a-good-emmcsolution-for-mobile

[13] S.Im和D。“组合：使用slc闪存提高mlc闪存的性能和寿命”，《系统架构杂志》，第1卷。56岁，没有。12，pp.641－653，2010.

[14] W.王，T。谢和D。周，“理解阈值电压对mlc闪存性能和可靠性的影响”，第28届国际会议论文集。*在超级计算*。ACM，2014年，页。201－210.

[15] Micron，“Mt29f32g08cba数据表”，2013年，访问日期： 20154-26。在线可用性： http://www.datasheetspdf.com/datasheet/mt29f32g08cba.html

[16] Y.胡，H。姜，D。冯，L。天，H。罗和S。张勇，“通过高级命令、分配策略和数据粒度对ssd并行性的性能影响和相互作用”，国际论文论文集。*在超级计算*。ACM，2011，pp。96－ 107.

[17] Micron，“Mt29f128g08cbcbb数据表”，2008，访问日期： 2015-4-26。在线可用性：
http://www.micron.com/～/media/documents/products/datasheet/nandflash/80series/185c_加_128gb_256gb_512gb_1tb_2tb_async_同步_nand.pdf

[18]——，“Mt29f64g08cbaa数据表”，2009年，访问日期： 2015-4-26。[在线条可用性： http://www.datasheetarchive.com/dl/Datasheets-IS23/ DSA00448784。pdf

[19] M.“安卓：改变移动领域》，普适计算，卷。10，没有。1，pp.4 - 7，2011.