|  |
| --- |
| **通过OPS隔离实现符合SLO要求的固态硬盘** |

|  |  |
| --- | --- |
| 院（系）名 称 ： | 计算机学院 |
| 学 生 姓 名 ： | 刘鑫瑞 |
| 学生学号： | 2017282110238 |

**摘要：**

虚拟化系统应该满足为每个虚拟机配置服务级别目标（SLO）。服务级别目标的表现，是通常由隔离虚拟机的底层硬件资源实现。在本文中，我们证实了通过评估SLO的性能不能满足商用固态硬盘的要求。我们发现了垃圾收集是这个问题的本源，而且不能轻易的解决，因为虚拟机之间存在交互。为了控制垃圾收集对虚拟机的影响，我们提出一个名为OPS隔离的方案。OPS隔离收集闪存块，以便在垃圾收集期间，一个虚拟机的块不会干扰其他虚拟机的块。实验结果表明，SLO的性能可以通过OPS隔离来实现。

**1介绍**

基于闪存的固态硬盘（SSD）现在已经十分常见，并且正在扩展到服务器虚拟化[1，2]。虚拟化系统对每个虚拟机应满足服务级别的目标（SLO）。服务级别目标表通常由隔离虚拟机之间的底层硬件资源实现。因此，许多关于分配虚拟机资源的研究已被实施，已经存在产品如VMware，ESX服务管理程序可以提供隔离的CPU和内存，并且已经可以使用[3，4]。一项最近的进行的研究是在虚拟化系统中使用SSD作为虚拟机之间共享的缓存[1,2]。在这项工作中，我们重新审视这个问题。首先，通过定量检查IO性能和虚拟机之间在SSD上的干扰，我们发现，根据SSD的状态，实验结果可以显著地变化，并且这个变化来自该干扰中的虚拟机。我们然后提出了在FTL（闪存转换层）上的OPS（配置空间上）隔离，因此每个虚拟机的配置空间都被隔离而不受其他虚拟机影响。我们的实验表明，虚拟机的服务级别目标性能可以是通过配置空间隔离来满足。

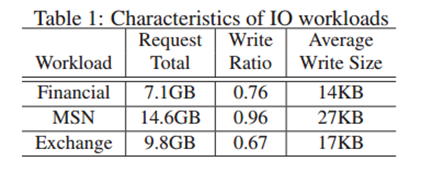
本文的其余部分安排如下。在该下一节，我们提出这项工作的动机和与这项研究有关的工作。在第3节，我们考虑SSD的内部以了解垃圾收集在并发虚拟机上的影响。在第4节中，我们提出OPS隔离和它的性能表现。最后，在第5节，我们给出一个总结。

**2动机和相关工作**

我们进行了一系列的实验并且观察了实验结果的表现。我们发现即使SSD执行相同的工作负载，它们的性能差别也很大。所以固态硬盘很容易受到其状态影响并且难以控制。该结果可作为开发性能可预测的SSD的动机。

本节中的所有实验均使用现成购买的商用SSD。该产品是基于MLC的128GB闪存。实验从一个早期干净状态或老化状态。老化是通过实施随机写入（包括覆盖）大小从4KB到32KB不等的数据，总写入量超过SSD容量。当SSD写满时，SSD将忙于执行垃圾收集。我们认为在此状态下的SSD是老化的SSD。

三个工作负载：Financial，MSN和Exchange被用于实验。表1显示了工作负载的细节特征。这里使用的原始工作负载是由UMassTraceRepository和MSR提供[5，6]。在本节的实验中使用真实的SSD，我们需要真实的IO请求。因此，我们使用一个替代工具从跟踪中获取请求并将其转换为真正设备请求[7]。对于每个请求，一个线程替换器等待它完成，在此基础上收集各种统计数据。



**2.1老化SSD的影响**

我们进行一系列的实验来表明虚拟机是如何的受存储系统各种条件的影响。首先，我们展示了随着SDD状态的变化这种影响的比例变化。我们的目标是在虚拟机之间按比例分配一个共享SSD的IO。要做到这样，我们首先创建基于内核的虚拟机（KVM），它们有相同的工作负载。虚拟机的设置和实验的其他环境在表2中显示。我们使用Cgroup[8] Linux特性限制并隔离硬件资源的使用情况，让小组为虚拟机分配不同的权重。然后我们测量每个虚拟机的IO性能。

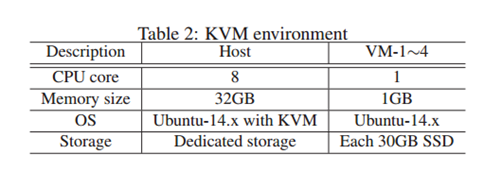
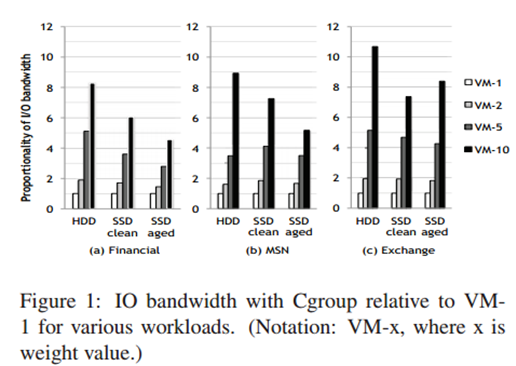
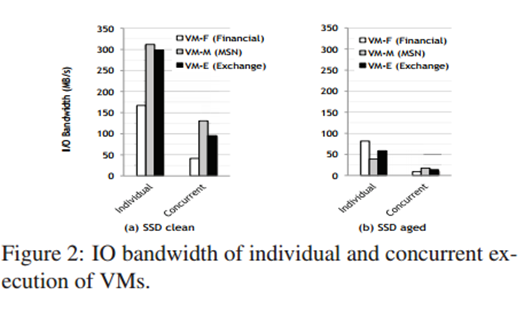


图1中的结果显示，对于所有工作负载，在HDD上比例接近于IO的权重，除了虚拟机10。但是，对于SSD来说，比例偏离较大。老化的SSD比干净的SSD表现的更糟糕。我们可以猜想，这是由于垃圾收集。事实上，我们稍后会展示在垃圾收集期间，虚拟机实际上正在移动其他虚拟机周围的数据，在垃圾收集触发的虚拟机观点上看这是不必要的数据移动，因而导致不精确的性能控制。从图1还可以观察出，垃圾收集对各种虚拟机的影响不同。也就是说，我们知道垃圾收集是有影响的，而且以一种负面的方式表现，但每个虚拟机是如何的受影响的不明确。



**2.2并行执行的影响**

我们进行另一组实验中，这次采用混合的工作负载。四组结果都呈现在图2中。图2（a）显示了一个干净SSD的带宽。单个的结果是通过在干净的SSD上执行一个工作负载获得的。并发的结果是通过同时在一个干净的SSD上执行三个工作负载获得的。这三个工作负载完全一样并且并行执行。



我们从结果中观察到并行执行的虚拟机性能明显比单个执行的虚拟机更差。并行执行时，每个虚拟机性能大约是单个执行时的三分之一，每个虚拟机之间有一些偏差。我们也观察到带宽没有被消耗完全占用，三个并行负载所消耗的带宽约为270MB/s。

图2（b）的实验与干净的SSD实验相似，只有SSD经历了一个老化的过程。这个结果中有三点值得注意。首先，总体表现下降是显着的，原因是垃圾收集。再次，我们也看到与单个执行相比，并行执行的性能下降是显著的，对于干净的固态硬盘，其下降的更多。最后，更重要的是老化的SSD对单个虚拟机的影响，而且根据虚拟机的不同而有很大不同。即，老化的影响是不统一的。例如，对于单个执行，而虚拟机F的带宽降到了虚拟机M带宽的一半，但在干净的SSD的情况下，只减少到15％。

同样，在其他小节我们得出了同样的结论。也就是说，我们指出了垃圾收集会降低性能。然而，该影响在单个虚拟机上并不完全一致。我们知道垃圾收集有负面影响，但虚拟机如何被影响还不明确。

**2.3相关工作**

在虚拟化系统中，服务级别目标（SLO）对于虚拟机来说，是通过透明地重新分配每个虚拟机的资源。诸如VMware，ESX之类的产品服务管理程序提供隔离的CPU和内存满足SLO[3,4]。许多研究已经实施以满足虚拟机的SLO[1,2,9,10,11,12,13,14]。特别地，DeepDive识别和管理共享硬件资源的虚拟机之间的性能干扰[11]。它被认为是第一个端到端的系统，处理主要的资源干扰，如CPU，内存和IO。

在虚拟机之间提供IO SLO的研究已经开始进行[9,10,14]。mClock提供通过IO在虚拟机之间按比例分配公平性管理程序的调度[10]。对分配虚拟机的共享固态盘缓存的研究也已被实施[1,2]。S-CAVE有效管理一个共享SSD通过在虚拟机之间使用运行时信息来缓存[1]。vCacheShare基于IO访问特性运行虚拟机解决了服务器的分配决策闪存缓存[2]。他们的工作的目标是在最大的调整SSD缓存的利用率并实现性能隔离。我们的工作表明，从外部控制SSD很难做到控制垃圾收集的内部运作。

最近的一项研究称为多流式SSD提出了和我们类似的方法[15]。这里，Kang等人建议改变区块设计，副接口来管理基于它们所调用的块流，也就是预期寿命相似的块。这个工作与我们的不同之处在于它们的重点最大限度地提高SSD的整体性能工作量独立的块表征，而我们集中精力控制SSD内的每个虚拟机符合SLO。

**3垃圾收集的影响**

在本节中，我们首先讨论垃圾收集对单个负载的影响，这是使用仿真环境完成的。然后我们列出实验结果，对于商用SSD也具有类似的效果。

**3.1垃圾收集对并发负载的影响**

为了分析并发工作负载的垃圾收集开销，我们用DiskSim的SSD扩展进行实验，其内部工作可以被监控。DiskSim采用大多数SSD产品中使用的页面映射FTL。至于垃圾收集，当空闲块的数量下降到低于阈值时，它使用贪婪的决策来选块。模拟器的其他参数列在表3中。此外，我们使用的工作负载与上一节的相同。图3显示了工作负载的访问模式。该图显示了每个负载获得的不同工作空间。

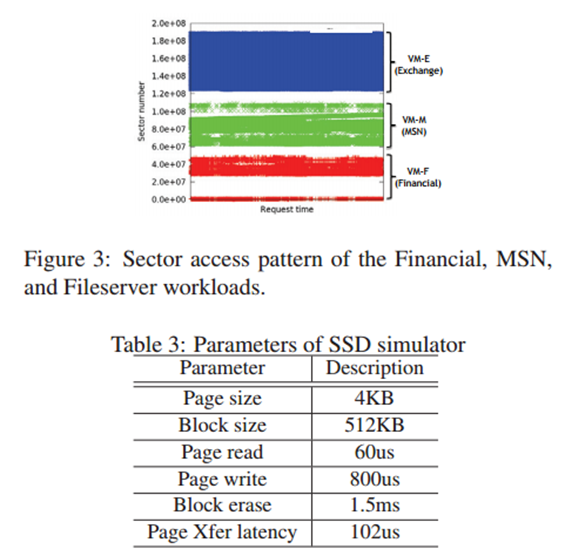
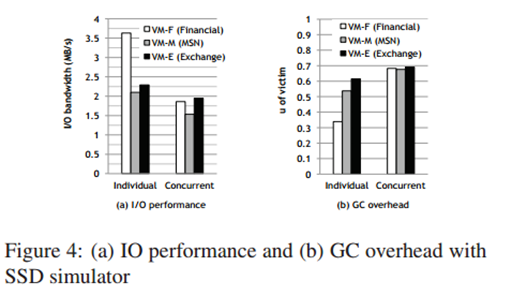


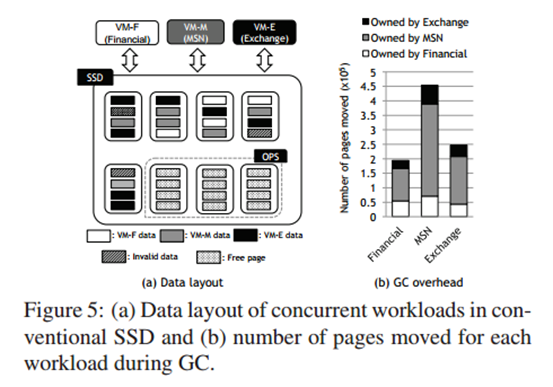
图4（a）显示的这种情况下的IO性能是单个工作负载下执行的，这和第二部分所描述的相似。在这里，测量性能之前，我们使用类似的方式使DiskSim模拟器老化。对于单个运行的情况下，性能的趋势类似于那些真正的SSD。在并发的情况下，该趋势是完全不同的，但是这是因为所使用的FTL将会与预期不同三个工作量不同程度地同时影响FTL。请注意，尽管确切的性能趋势可能会有所不同，单个负载下的性能下降的变化与商用SSD观察到的一样。



现在我们来看看这个观察背后的原因。为此，我们观察内部的状态和设备内页的移动。这是通过在页面做标记，其ID为虚拟机启动请求和监视的的标签。图4（b）显示了有效页面的平均数量表示为u，对于垃圾收集的所选块。这个值较低，比工作负载执行时要单独进行同时。特别的，u和图4（b）中的性能是成比例的。这就是说，虽然每个工作负载在执行时都受到负面影响，但Financial受到最大的影响。

这种负面影响背后的原因可以通过图（5）来解释。图5（a）列出了SSD的数据输出，当来自多个工作负载的请求同时到达时。该FTL每个页和将它们随机放置在可用块中。同时块中包含来自各种工作负载的页面。因此，在擦除同时执行一个特定的工作负载时，加载来自所选块中其他工作负载的实时页面将在垃圾收集过程中移至新块。

图5（b）显示了在每个工作负载的垃圾收集期间被移动的页面数量。当它们被移动时，这些页面将被其所有者抛弃。例如，移动了190K页的工作量，只有30％属于它自己。这说明，虽然垃圾收集是必须的，但其中涉及的大部分工作实际上是其他负载导致的不必要的工作。然后解决这个问题的办法是找到一种方法来隔离这个垃圾收集进程，以便每个工作负载的垃圾收集不受其他工作负载干扰。

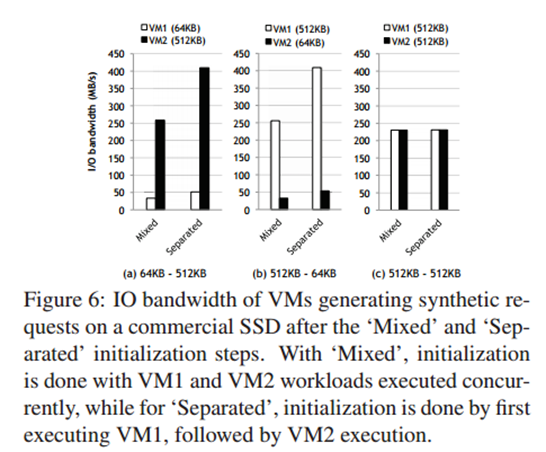


**3.2商用SSD上的观测**

在前面的小节中，我们展示了用DiskSim解决干扰现象的仿真环境。虽然模拟是许多重要的研究和创新的基础，但还是不得不怀疑我们在前面小节中所观察到的是否实际发生在真正的SSD中。为了验证这一点，我们执行以下一组实验。用我们以前使用的商用SSD，我们创建两个虚拟机，虚拟机1和虚拟机2生成写入并重写64KB和512KB大小的随机数据，分别代表小的和大的请求的任务。两种尺寸的选择是改变混合的内的数据块作为将被描述在下面。因此，在我们显示的结果不会因不同的尺寸而有所不同，只要这两种尺寸有一些显著的差异即可。

有了这两个虚拟机，我们执行两组不同的实验。第一组实验，开始从一个干净的SSD，在两个虚拟机作为初始化步骤同时运行一些时间。作为结果，该SSD会弹出用来自虚拟机1和虚拟机2的数据导致两个虚拟机的数据混合。然后，该虚拟机再次运行，但这次的表现在图6（a）中被表示为“混合”。第二组的实验中，我们也经历了一个初始化过程但这次虚拟机一次只能运行一个虚拟机相同量的数据。这一次因为请求的大小和虚拟机的顺序运行，FTL将不会把两个虚拟机的数据放置一块。然后，在虚拟机运行并测量性能。这些结果在图6（a）中被表示为“分离”。需要注意的是“分离”比“混合”好得多。

图6（b）中显示的结果是来自同样的实验，只是初始化时先用512KB，然后64KB的随机写入。图6（c）显示了对于相同的实验序列，但具有相同的大小（512KB）随机写入的实验结果。这些结果显示与图6（a）和（b）的对比。



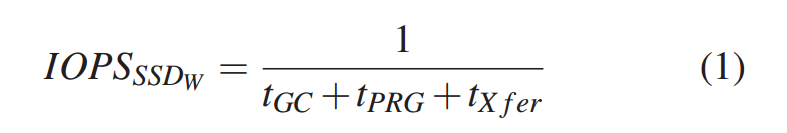
“分离”比“混合”性能好的原因可能是在来自一个虚拟机的页面的“分离”场景下，不会影响其他虚拟机。虽然我们不能确定在SSD内的工作原理，但这些结果与DiskSim评估中的发现是一致的的。

**4 SLO兼容SSD及其性能评估**

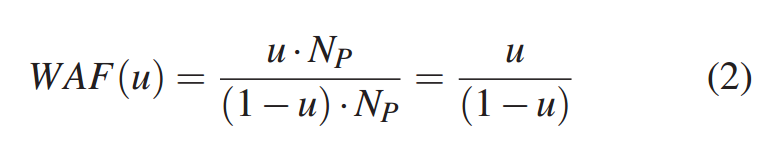
在本节中，我们将讨论如何将减轻垃圾收集的影响，以便SLO请求可能是满意的。我们从回顾以前的基于闪存的IO性能固态硬盘工作开始。我们提出将OPS隔离作为控制手段各个虚拟机的性能。实验结果表明虚拟机的性能可以通过OPS隔离得到提高。

**4.1计算SSD的IOPS**

为了保证共享一个SSD虚拟机的IO性能，我们需要了解IO之间的关系性能和垃圾收集开销。NAND闪存固态硬盘的性能特性已经有了广泛的研究，并从这些可以很好理解的写IO性能，如图所示表示在公式1中[16,17]。



WAF的计算如公式2。需要注意的是u可以从该SSD被测量或者可以根据用户数据的比例来估算和最初的OPS规模[18,19]。另外请注意，u是一个代表整个SSD的值。

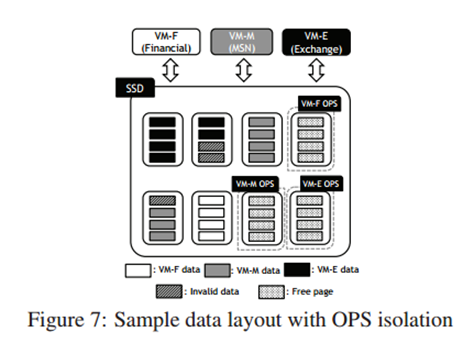


从公式1和公式2，我们知道固态硬盘写操作的性能由垃圾收集开销决定，u是由OPS确定的[20]。因此，为了控制IO性能，适当操作OPS是必要的。

通常，OPS在SSD中是全局管理的。因此，基于虚拟机的IO性能保证是困难的，否则不可能处理。我们建议手动隔离OPS，以便每个虚拟机能管理OPS。这允许每个虚拟机在IO控制上有更多的控制。

**4.2 OPS隔离**

为了满足虚拟机的SLO请求，我们建议给每个虚拟机分配闪存块包括OPS。当给虚拟机分配页时，可以预防垃圾收集时的干扰。图7显示了这三个区块将如何分配虚拟机同时请求闪存空间。对比图5（a）的这个数据显示了这两者之间的关系不同。在图7中观察，所有的块都包括自由页，或者只有一个虚拟机的页面。由于OPS也是属于单个虚拟机，每个虚拟机写入请求将只能放在同一块，防止来自不同的虚拟机的页混合。



为了满足SLO请求，IO性能也必须保证。正如4.1节所讨论的，性能最终受到分配给每个虚拟机的OPS的影响。对于这项研究，我们只是采取比例划分的为了满足SLO请求的总可能的IOP。

最初，闪存块，包括OPS，根据权重被划分在竞争的虚拟机之间。使用该IOPS限定SSD，我们根据每个虚拟机计算估计的IOPS要求的权重。然后，用公式1可以计算出每个虚拟机的u。初始化之后，OPS大小是动态并定期进行调整以保持当前的IOPS初始指定。例如，以3个虚拟机A，B，C为目标对象，1，3和6的IO性能权重更高权重被分配更高的带宽。最初的OPS每个虚拟机的大小由权重的比率来设置3和4，即OPS的10%，30%和60%分别分配给虚拟机A，B，C。如果我们假设该指定的IOPS为1000，即目标IOPS也由每个虚拟机指定的权重。最后，使用公式1将为每个虚拟机设定利用率。然后，如果目标利用率变化，则可以调整OPS大小。这个调整是在之前完成的每个与OPS大小增加或减少的垃圾收集一个必要的块。为了在SSD中实现这样的功能，存储接口必须改变。最近的研究，例如多流SSD[15]已经提出了改变增强性能优势的界面。同样地，我们的方法需要更少的信息被转移到SSD，例如标识一个已工作的负载，它已被与eMMC闪存提供[21]，以及SLO要求的权重。

**4.3性能评估**

为了评估我们提出的符合SLO的SSD，我们在DiskSim SSD [22]上进行实验，这在第3节中已被使用。我们再次使用以前使用的工作负载。

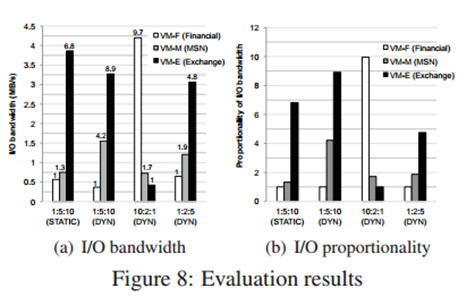
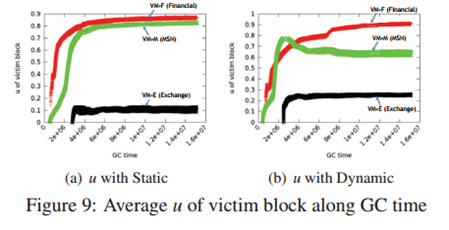


图8（a）显示了虚拟机权重不同时的各种工作负载的结果。图中，x轴表示执行的虚拟机组同时分配给虚拟机的权重。对于在静态情况下，OPS的大小并没有改变整个执行。对于结果的其余部分，将OPS尺寸会动态调整。y轴表示实现的绝对带宽和每个小节顶部的数字代表性能相对于最小重量的比例。为了便于比较，图8（b）给出和图（1）相同的结果格式。

结果显示，使用OPS分离和动态调整OPS大小，对于IO带宽的比例相当准确。但是，静态OPS隔离并不高效是因为无法根据工作量调整OPS大小。图9（a）显示了当OPS被设置为一个静态值时u的变化。作为对比，图9（b）显示了根据需求动态调整OPS大小时u的变化。



**5结论**

在本文中，我们证实了SLO性能不能满足目前的商用SSD，因为垃圾收集会干扰竞争的虚拟机。为了解决这个问题，我们提出了OPS隔离，分配闪存以阻止块之间不共享虚拟机，但是完全专用于每个虚拟机。我们的实验结果表明，OPS隔离是一个为竞争的虚拟的SSD提供SLO性能的有效方式。

**参考：**

[1] Tian Luo, Siyuan Ma, Rubao Lee, Xiaodong Zhang, Deng Liu, and Li Zhou. S-CAVE: Effective SSD Caching to Improve Virtual Machine Storage Performance. In Proc. of International Conference on Parallel Architectures and Compilation Techniques (PACT), pages 103–112, 2013.

[2] Fei Meng, Li Zhou, Xiaosong Ma, Sandeep Uttamchandani, and Deng Liu. vCacheShare: Automated Server Flash Cache Space Management in a Virtualization Environment. In Proc. Of USENIX Conference on USENIX Annual Technical Conference (ATC), pages 133–144, 2014.

[3] VMware Inc. Distributed Resource Scheduler. http://www.vmware.com/files/pdf/VMware-Distributed-Resource-Scheduler-DRS-DS-EN.pdf.

[4] Carl A. Waldspurger. Memory Resource Management in VMware ESX Server. SIGOPS Oper. Syst. Rev., 36(SI):181–194, 2002.

[5] UMASS TRACE REPOSITORY. OLTP Application I/O. http://traces.cs.umass.edu, 2002.

[6] Dushyanth Narayanan, Eno Thereska, Austin Donnelly, Sameh Elnikety, and Antony Rowstron. Migrating Server Storage to SSDs: Analysis of Tradeoffs. In Proc. of ACM European Conference on Computer Systems (EuroSys), pages 145–158, 2009.

[7] Yongseok Oh. Trace-replay. https://bitbucket.org/yongseokoh/trace-replay.

[8] Paul Menage. CGROUPS.

https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt.

[9] Ajay Gulati, Irfan Ahmad, and Carl A. Waldspurger. PARDA: Proportional Allocation of Resources for Distributed Storage Access. In Proc. of USENIX Conference on File and Storage Technologies (FAST), pages 85–98, 2009.

[10] Ajay Gulati, Arif Merchant, and Peter J. Varman. mClock: Handling Throughput Variability for Hypervisor IO Scheduling. In Proc. of USENIX Conference on Operating Systems Design and Implementation (OSDI), pages 1–7, 2010.

[11] Dejan Novakovic, Nedeljko Vasi ´ c, Stanko Novakovi ´ c, Dejan ´ Kostic, and Ricardo Bianchini. DeepDive: Transparently Identi- ´ fying and Managing Performance Interference in Virtualized Environments. In Proc. of USENIX Conference on Annual Technical Conference (ATC), pages 219–230, 2013.

[12] R. Prabhakar, S. Srikantaiah, C. Patrick, and M. Kandemir. Dynamic Storage Cache Allocation in Multi-Server Architectures. In Proc. of Conference on High Performance Computing Networking, Storage and Analysis (SC), pages 8:1–8:12, 2009.

[13] David Shue, Michael J. Freedman, and Anees Shaikh. Performance Isolation and Fairness for Multi-tenant Cloud Storage. In Proc. of USENIX Conference on Operating Systems Design and Implementation (OSDI), pages 349–362, 2012

[14] Eno Thereska, Hitesh Ballani, Greg O’Shea, Thomas Karagiannis, Antony Rowstron, Tom Talpey, Richard Black, and Timothy Zhu. IOFlow: A Software-defined Storage Architecture. In Proc. of ACM Symposium on Operating Systems Principles (SOSP), pages 182–196, 2013.

[15] Jeong-Uk Kang, Jeeseok Hyun, Hyunjoo Maeng, and Sangyeun Cho. The Multi-streamed Solid-State Drive. In USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage), 2014.

[16] Xiao-Yu Hu, Evangelos Eleftheriou, Robert Haas, Ilias Iliadis, and Roman Pletka. Write Amplification Analysis in Flash-based Solid State Drives. In Proc. ACM International Ststems and Storage Conference (SYSTOR), pages 10:1–10:9, 2009.

[17] Yongseok Oh, Jongmoo Choi, Donghee Lee, and Sam H. Noh. Caching Less for Better Performance: Balancing Cache Size and Update Cost of Flash Memory Cache in Hybrid Storage Systems. In Proc. of USENIX Conference on File and Storage Technologies (FAST), pages 313–326, 2012.

[18] Hunki Kwon, Eunsam Kim, Jongmoo Choi, Donghee Lee, and Sam H. Noh. Janus-FTL: Finding the Optimal Point on the Spectrum between Page and Block Mapping Schemes. In Proc. International Conference on Embedded Software (EMSOFT), pages 169–178, 2010.

[19] Wenguang Wang, Yanping Zhao, and Rick Bunt. HyLog: A High Performance Approach to Managing Disk Layout. In Proc. Of USENIX Conference on File and Storage Technologies (FAST), pages 145–158, 2004.

[20] Radu Stoica and Anastasia Ailamaki. Improving Flash Write Performance by Using Update Frequency. Proc. VLDB Endowment, 6(9):733–744, 2013.

[21] JEDEC. Data Tag Mechanism of eMMC, JEDEC Standard Specification No. JESD84-B45.

[22] Vijayan Prabhakaran and Ted Wobber. SSD Extension for DiskSim Simulation Environment. http://research.microsoft.com/en-us/downloads/b41019e2-1d2b-44d8-b512-ba35ab814cd4, 2009.