一种固态盘的读写性能优化调度方法

朱玥[1] 吴非[1,2] 熊钦[1] 谢长生[1,2]

[1]（武汉光电国家实验室 武汉 430074）
[2]（信息存储系统教育部重点实验室 武汉 430074）
CONTACT: wufei@hust.edu.cn

ABSTRACT

相比传统机械硬盘，基于NAND Flash的固态盘具有非易失性、高性能、低功耗等优点，因此被广泛应用于数据中心、云计算、在线事务交易等场景。然而，由于NAND Flash中读操作速度远远高于写操作速度，当读写请求并发执行时，读请求可能被写请求所阻塞，从而表现出极大的读延时。在许多以读请求为主的场合，尤其是在线事物交易中（读请求占总请求的比例超过90%），读延时的急剧增加更是严重影响了系统的整体性能。

本文提出一种读写性能优化调度的策略，通过动态调整读写请求的优先序列，使读性能显著提升。并且，在固态盘仿真平台下实现了对读写调度策略有效性的系统评估。实验结果表明，在该调度策略下，系统中读延时的最大值和平均值均得到了显著的减少，平均降幅分别达到了72%和41%。

CONTRIBUTIONS

- ✓ **降低读延时：**通过提升读请求的优先级，减少写请求对读请求的阻塞，有效降低读延时，提高了读性能。
- ✓ **提升系统性能：**在显著提升读性能的同时，写性能仅存在轻微下降，甚至在许多情况下略有提升，从而在整体上提升了系统性能。

PROBLEM FORMULATION

- ✓ NAND Flash的两个特性：

（1）读写不对称：在NAND Flash中，写操作的时间远大于读操作的时间；

（2）Die是接收和执行Flash命令的基本单元：在一个Die中，当对某个页进行写操作的同时，不能对其它页进行读操作。

- ✓ **导致的问题：**

当请求的目标地址在同一个Die中时，读请求可能被写请求所阻塞，从而造成较大的读延时，严重影响系统的读性能。

SCHEDULING SCHEME

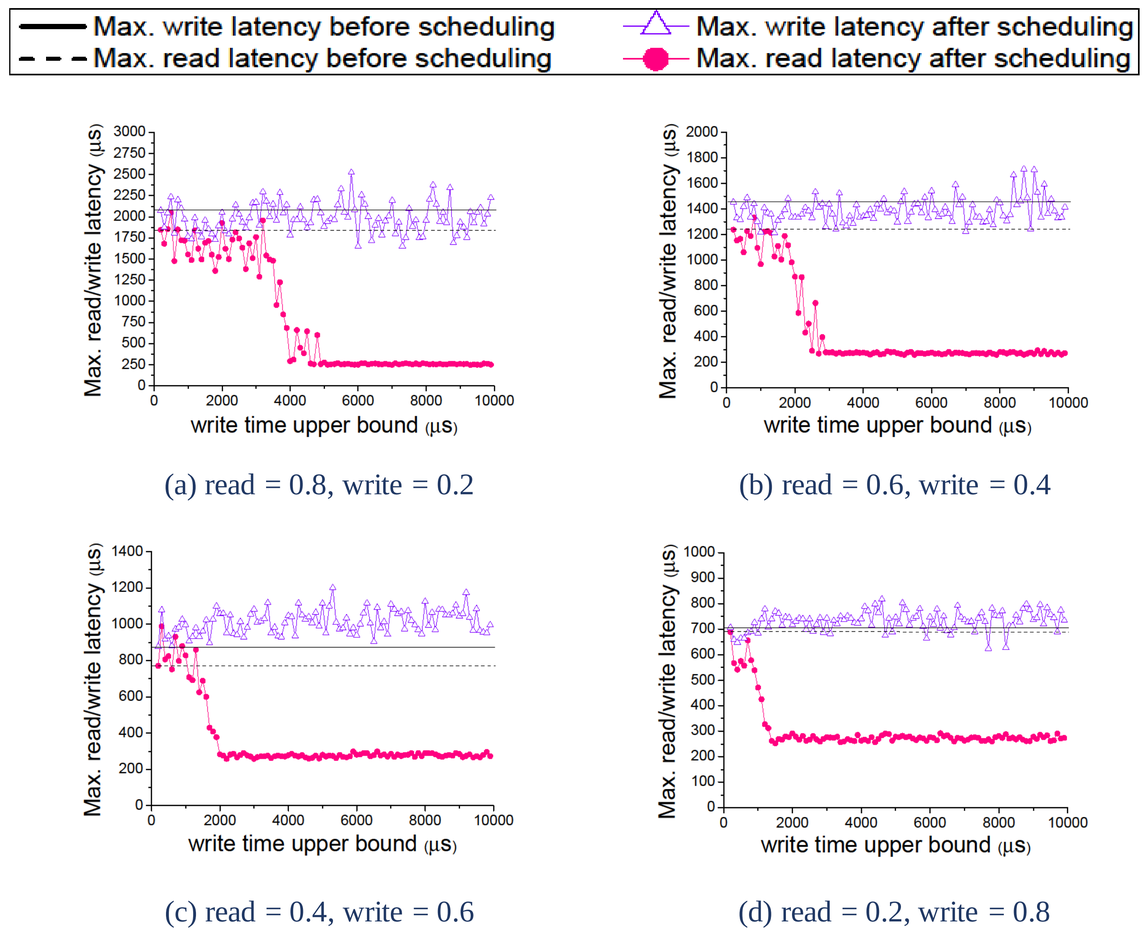
由于读操作和写操作的时间存在着一个数量级的倍数关系，因此，对于被写请求所阻塞的读请求来说，读延时可能达到不被阻塞时的上十倍；相反，若优先执行读请求而将写请求推后，写延时只有轻微的增长。

本文提出一种读写调度方法，通过适当提升读请求的优先级，推后执行写请求，从而以轻微牺牲写性能为代价，显著提升读性能，进而有效改善系统的整体性能。

Fig.1 Improving read performance by scheduling

- ✓ **读请求尽可能提前：**赋予读请求较高的执行优先级，可以通过轻微的牺牲写性能，换取读性能的显著提升。
- ✓ **写延时尽量不超过设定的上限：**为写延时设置上限（write time upper bound），一旦超过上限，该写请求不能被队列中位于其后的读请求优先执行，从而避免写请求被不断推后，造成写饥饿的现象。
- ✓ **若存在读写相关性则先写后读：**当一个读请求和一个写请求的目标地址相同（为同一个页）时，称这两个请求具有读写相关性；如果在队列中，一个读请求之前存在与之具有相关性的写请求，那么需要遵循先写后读的原则，以避免造成读数据错误。
- ✓ **相同类型请求之间的位置关系不变：**为保证公平性，在调度过程中，两个相同类型请求之间的位置关系不能够发生改变。

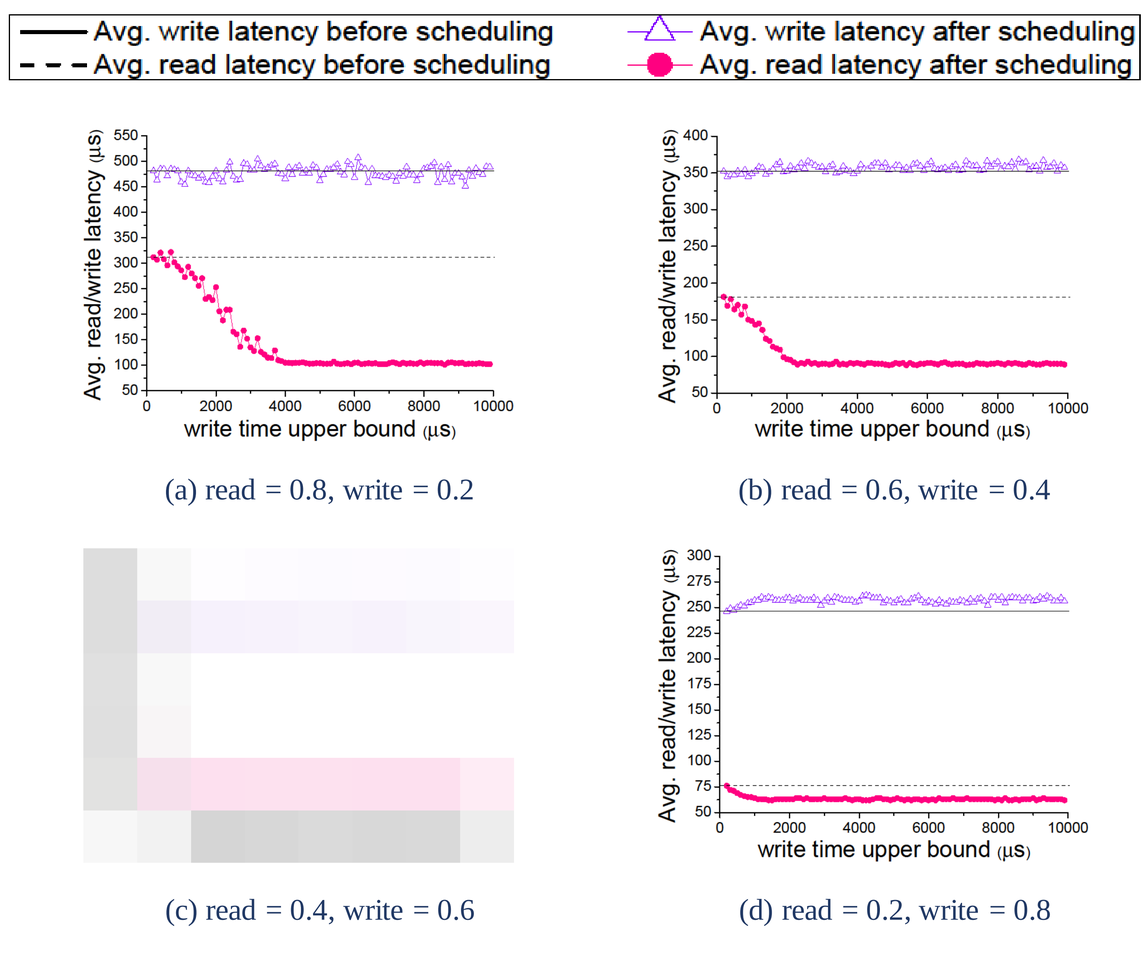
SIMULATION



(a) read = 0.8, write = 0.2　(b) read = 0.6, write = 0.4

(c) read = 0.4, write = 0.6　(d) read = 0.2, write = 0.8

Fig.2 Changes of the maximum read/write latency with write time upper bound when the ratios of read requests equal {0.8, 0.6, 0.4, 0.2}, with read latencies reduced by {86%, 78%, 64%, 61%} and write latencies increased by {-4%, -4%, 12%, 2%}, respectively



(a) read = 0.8, write = 0.2　(b) read = 0.6, write = 0.4

(c) read = 0.4, write = 0.6　(d) read = 0.2, write = 0.8

Fig.3 Changes of the average read/write latency with write time upper bound when the ratios of read requests equal {0.8, 0.6, 0.4, 0.2}, with read latencies reduced by {65%, 50%, 32%, 17%} and write latencies increased by {1%, 2%, 5%, 4%}, respectively

CONCLUSIONS

- ✓ 通过适当提高读请求的优先级，推后执行耗时的写操作，以显著提升读性能，并有效改善系统的整体性能。
- ✓ 本文提出的调度策略保障了在调度过程中，写请求不会被不断推后，读数据不因先读后写而产生错误，以及两个相同类型请求之间的公平性。
- ✓ 仿真结果表明，该调度策略能够显著提升系统的读性能，并且与此同时写性能只有轻微的降低，甚至在许多情况下略有提升，从而验证了该调度策略的有效性。