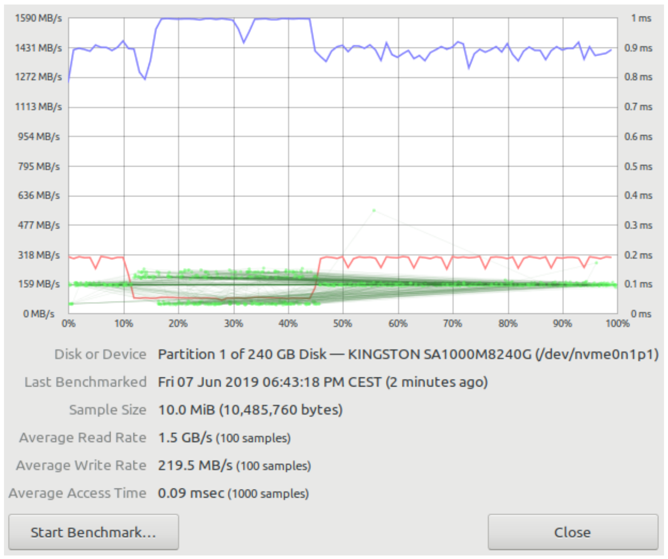
Computational Geometry (2021)
I/O-Efficient Algorithms

Peyman Afshani

October 25, 2021

Question 1 (The Flash Memory Model.): By today's standards, hard disks are considered an old technology and they are being replaced by Solid State Disks (SSDs), at least in the consumer product market. However, the internal structure of SSDs is very complicated, specially when it comes to how they deal with *write* operations and another curious aspect of them is the asymmetry between reads and writes where reads are generally much faster and "cheaper" than writes (see Figure 1)



Disk or Device Partition 1 of 240 GB Disk — KINGSTON SA1000M8240G (/dev/nvme0n1p1)
Last Benchmarked Fri 07 Jun 2019 06:43:18 PM CEST (2 minutes ago)
Sample Size 10.0 MiB (10,485,760 bytes)
Average Read Rate 1.5 GB/s (100 samples)
Average Write Rate 219.5 MB/s (100 samples)
Average Access Time 0.09 msec (1000 samples)

Figure 1: The test of my own SSD when I bought it in 2019.

Operation	**Speed**
Sequential Read	1500 MB/s
Random Read	123 MB/s
Sequential Write	750 MB/s
Random Write	23 MB/s

Table 1: My own (probably more accurate) benchmark of my SSD; the sequential Read/Write speed is what is being advertised by the producing company.

A very short summary is that while SSDs allow fast reads and writes, it is still true that accessing a *block* of elements is faster than accessing individual elements scattered all over the SSD (pay attention to the differences between sequential and random accesses in Table 1). Furthermore, *write* operations are much more costly than *read* operations and in fact *write* operations "wear down" the device; a major part of the internal complexity of SSDs is that they try to avoid writing the same physical location, to avoid hardware failures.

Modelling SSDs. Motivated by this, we try to adapt the I/O model of computation to SSDs. Our replacement model is something like this: we assume the internal memory has size M, and we have a SSD that is divided into blocks of size B (some theoretical models even go further and consider different block sizes for read and writes; we keep things simple here). However, we would like to distinguish between reads and writes. This means that whenever we analyze an algorithm, we give one bound for the number of reads and another bound for the number of writes. And during the algorithm design process, our goal is mostly to minimize the *write* operations.

Now, consider the permutation problem: we have an input of N elements and we would like to permute them into a permutation π. For simplicity, assume that $B > \log_{M/B} N$. Show that there exists a constant $\varepsilon > 0$ such that any algorithm that can solve the permutation problem with at most $\varepsilon \frac{N}{B} \log_{M/B} N$ write I/Os must have $\Omega(N)$ reads in the worst-case.

This result shows that it is not really possible to reduce the number of writes without blowing up the number of reads by almost a factor B.

(Hint: Perhaps you need to review the permutation lower bound proof for the I/O model).

Question 2 (Permuting and Sorting in SSDs):

- Part A (random permutation). Consider an input set S of N elements on an SSD disk. Assume that we are also given a random permutation, π, of the integers 1 to N on the SDD disk. Show that it is possible to create a random permutation of S using $O(N)$ read I/Os and $O(N/B)$ write I/Os.

- Part B (very write-optimized sorting). Consider and input S of N comparable elements on an SDD disk. Show that it is possible to sort S using $O\left(\frac{N^2}{MB}\right)$ read I/Os such that every element is written exactly once (here we assume $N \geq MB$).

- Part C (optional). Show that the number of reads can be reduced to $O\left(\frac{N^{4/3}}{MB} + N \log N\right)$ by allowing most elements to be written at most twice; in particular, all but some $O(N^{2/3})$ elements are written twice and those $O(N^{2/3})$ elements are written three times (here we assume $N^{1/3} \geq MB$).

Question 3 (rectangle enclosure): Design an $O(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{B} + \frac{T}{B})$ I/O algorithm for the following problem: Given N axis-parallel rectangles and N points in the plane, compute for each point p all the rectangles that contain p. Here T is the total number of rectangles reported.

(Hint: Use distribution sweeping with the multi-slab idea.)