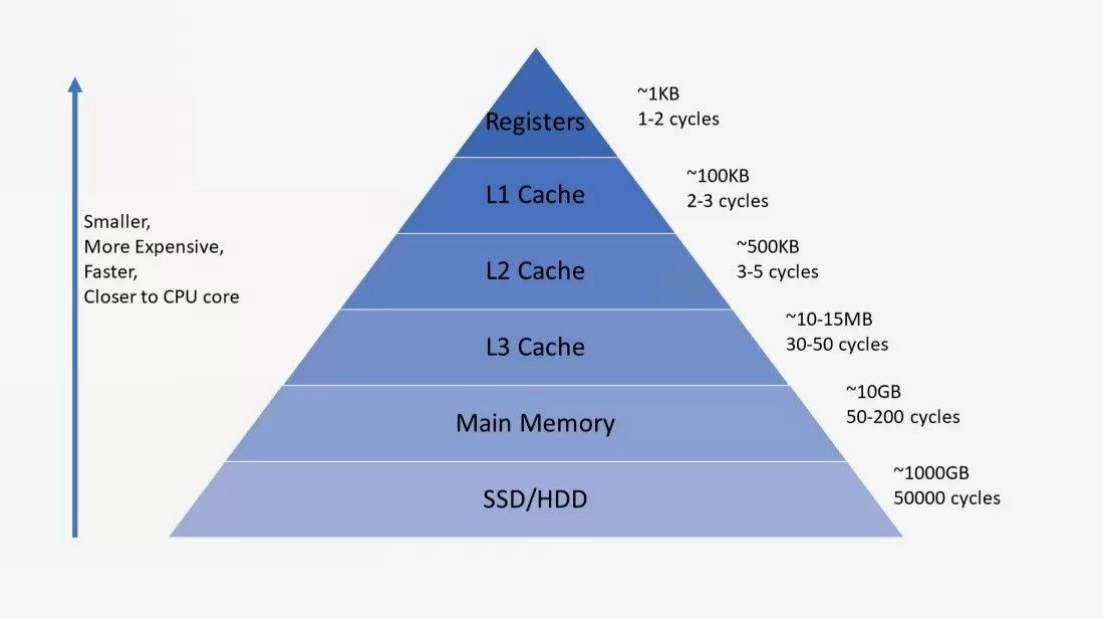
External Sorting Algorithms

Authors: Alok Aggarwal and Jeffrey Scott Vitter

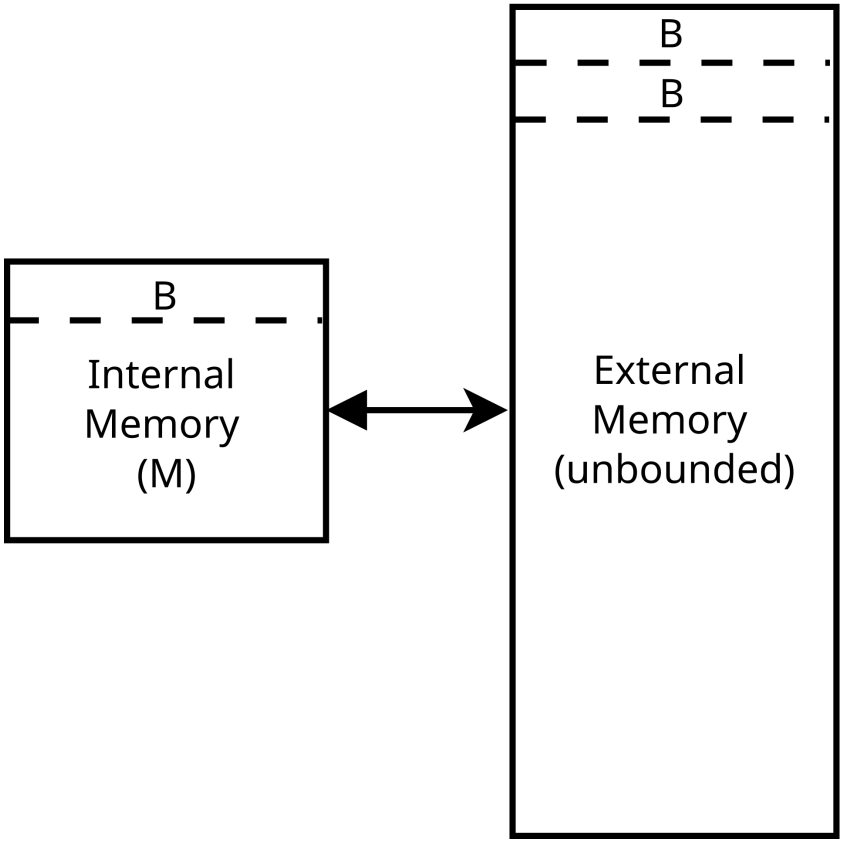
Background

What happens when our data doesn’t fit in RAM?



Model

We will assume an idealized model of computation called the “external memory model”.



Parameters

N: size of data

M: size of memory

B: block size (unit of I/O)

P: I/O parallelism

We will always assume **N** >> **M**.

Performance Analysis

In external memory algorithms, runtime is typically dominated by I/O (recall a disk seek is orders of magnitude slower than a RAM access!).

So, to optimize runtime performance, it usually suffices to minimize the number of I/Os.

External Sorting

Result:

The average-case and worst-case number of I/Os required is

$$\Theta\left(\frac{N}{PB} \cdot \frac{\log(N/B)}{\log(M/B)}\right)$$

Furthermore, this bound is asymptotically tight.

External Sorting Algorithms

We consider two different sorting algorithms that achieve this bound.

1. External merge sort:
    - sort phase + merge phase

2. External distribution sort:
    - divide and conquer

Applications

Although this paper is purely theoretical and does not present any experimental results, these algorithms that are aware of the memory hierarchy are still important.

In particular, there’s nothing specific to RAM/disk in these algorithms, and they can be applied to any situation where we have programmatic control over I/O, e.g., shared memory on GPUs.

Future Directions

1. Modern disk hardware looks quite different to the simplistic external memory model. How can we model it more realistically?
2. What other problems can we prove theoretical upper/lower bounds for in the external memory model. The paper discusses a few more, such as permutation, matrix transposition, and FFT digraph.