External Memory Algorithms

Chițaniuc Mirela

Introduction

- As problem data sets become larger and larger, it is often the case that problems do not fit into computer's internal memory.

- Standard virtual memory helps us store the rest of the problem to disk but this technique is quite slow because of the bottleneck between disk and internal memory.

- In the paper, the approach proposed in order to solve such problems is EM algorithms/data structures that want to reduce I/O costs and exploit data locality.

Introduction

- We need to work with a lot of data so multiple parallel disks are considered here.
- A number of techniques are considered as well for parallel disk computation as we will see in the next sections.
- Some examples are *disk striping*, *distribution*, *merging techniques*, etc.
- Some analysis are done on diferent types of problems, sorting, graph problems, geometric problems, etc.
- We will present a number of problems and the results obtained, although we cannot cover all of them.

Background

- Modern architectures are reliant on a hierarchical memory structure.

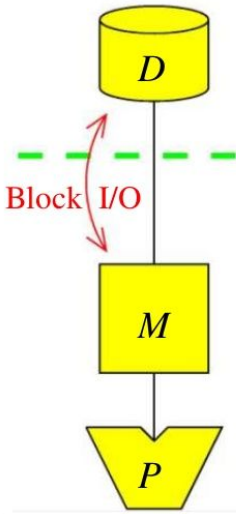
- Things like virtual memory, that lets us extend the capabilities of the regular internal memory and other techniques such as caching and prefetching are very practical for general purpose applications.

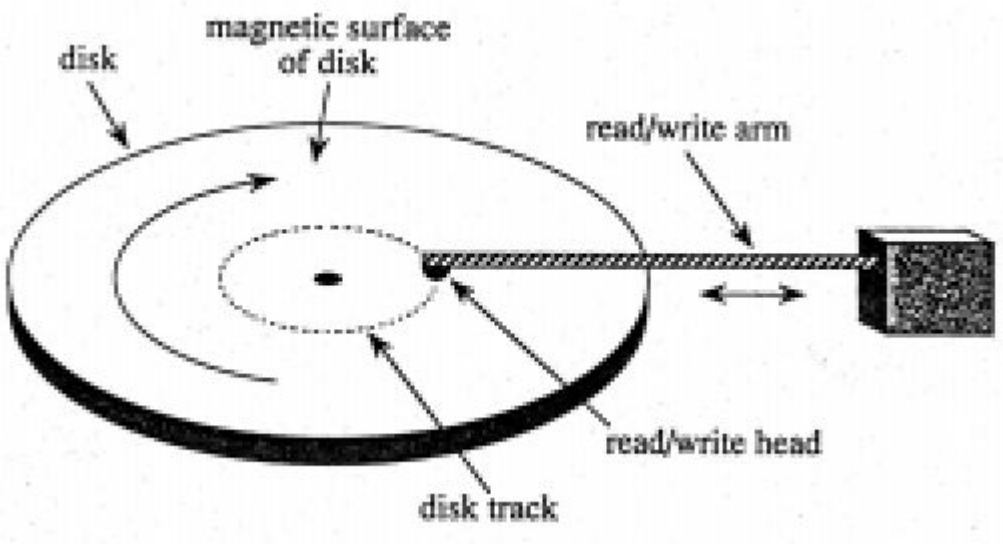
- But they are build with the thought of generality in mind. So they cannot take full advantage of other things such as data locality which is very important.

External Memory Algorithms

- Thus we can see if by using explicit memory management of contents on each level in the hierarchy we can achieve a better performance.
- This types of algorithms/data structures that explicitly manage data placement and movement are called EM algorithms and data structures.
- For simplicity and without loss of generality we only discuss only the I/O communications between disk and internal memory.
- EM algorithms and data structures are often designed and analyzed using the parallel disk model (PDM)

Parallel disk model



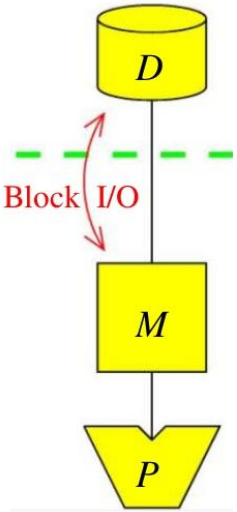


Parallel disk model

- Magnetic disks consist of one or more rotating platters and one read/write head per platter surface. The data are stored in concentric circles on the platters called **tracks**.
- To read or write a data item at a certain address on disk, the read/write head must mechanically seek to the correct track and then wait for the desired address to pass by.
- The seek time to move from one random track to another is often on the order of 5-10 milliseconds, and the average rotational latency, which is the time for half a revolution, has the same order of magnitude.
- In order to amortize this delay, it pays to transfer a large collection of contiguous data items, called a **block**.

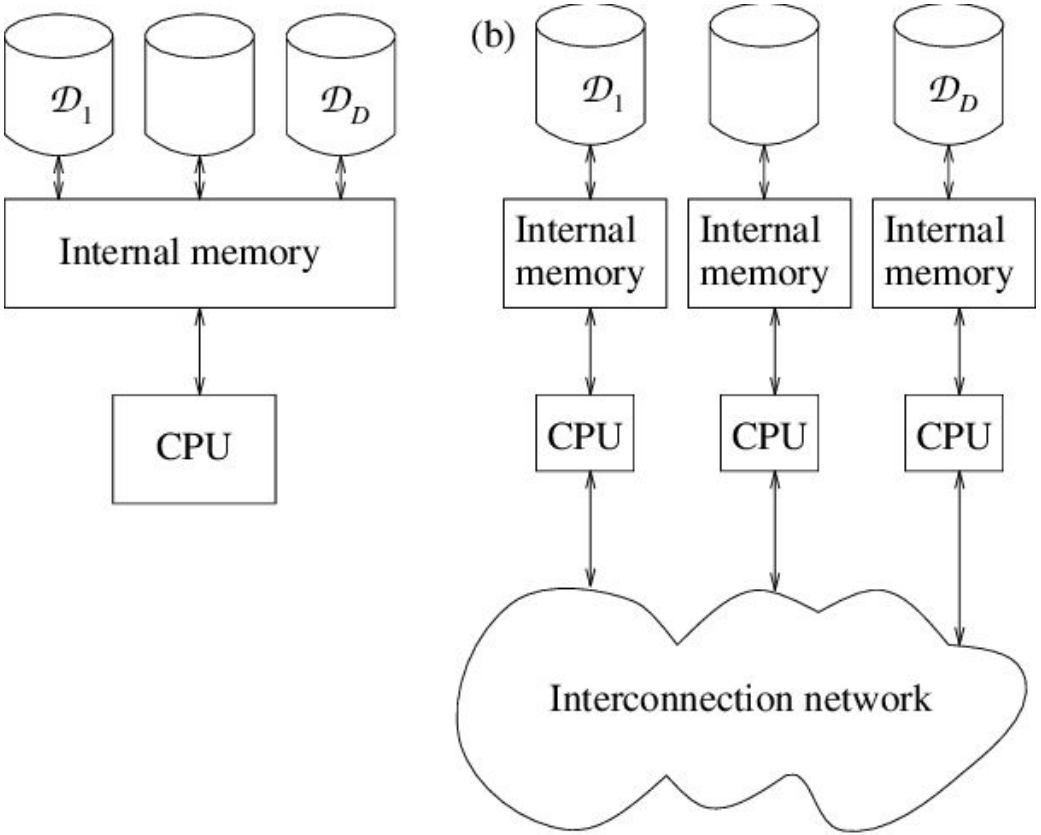
Parallel disk model

- We need to know that data is stored on disks on tracks and data is grouped in blocks.
- In a single I/O operation we can read/write a single block of data.



- The problem parameters that we use in computing a performance metric are :
  - N - problem size (in units of data items)
  - M - internal memory size
  - B - block transfer size
  - D - number of independent disks
  - P - number of CPUs
  - With M < N and 1 <= DB <= M/2

Parallel disk model



Performance in PDM

- For measuring performance in PDM we use:
  - ❖ the number of I/O operations (1)
  - ❖ the amount of disk usage (2)
  - ❖ internal computation time (3)
- In the article they only use (1) and (2).

Fundamental Bounds

	Internal	External
Scanning:	N	$\frac{N}{B}$
Sorting:	$N \log N$	$\frac{N}{B} \log_{M/B} \frac{N}{B}$
Permuting	N	$\min\{N, \frac{N}{B} \log_{M/B} \frac{N}{B}\}$
Searching:	$\log_2 N$	$\log_B N$

Locality and load balancing

- As we mentioned before, we need to take full advantage of data locality in order to design very good EM applications.
- When having a multiple disk model, the locality is more akin to a load balancing of the data because the data is spread on multiple disks.
- To put things into perspective we can look at the D = 1 special case to see how an algorithm without any optimization, I/O wise, can perform.
- For an algorithm that runs in O(NlogN) time, we use Ω(Nlog (N/B)) I/O operations. Considering that disk reads are slow, this is a major bottleneck.

Disk striping

- Disk striping is a paradigm that can ease the programming task with multiple disks.
- We assume that the data on all disks is stripped, meaning that the disks are viewed as a single large disk in which the data is stored as in the table below.

	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$
stripe 0	0 1	2 3	4 5	6 7	8 9
stripe 1	10 11	12 13	14 15	16 17	18 19
stripe 2	20 21	22 23	24 25	26 27	28 29
stripe 3	30 31	32 33	34 35	36 37	38 39

- Striped format allows a file of N data items to be read or written in *O(N/DB)* = *O(n/D) I/Os*, which is optimal.

Disk striping and sorting

- Disk striping on multiple disks can improve many EM algorithms when used properly but there is a case in which disk striping is not such a good idea.

- Sorting is such an example. We know when a sorting is optimal, and that is when it done using $\Theta\left(n\frac{\log n}{\log m}\right) = \Theta\left(\frac{N}{B}\frac{\log(N/B)}{\log(M/B)}\right)$ on a single disk.

Disk striping and sorting

- When using disk striping, we know that the disks are seen as a single disk with block size DB. Then the I/O bound is $\Theta\left(\frac{N}{DB}\frac{log(\frac{N}{DB})}{log(\frac{M}{DB})}\right)$
- But we know that the optimal bound for sorting is $\Theta\left(\frac{N}{DB}\frac{log(\frac{N}{B})}{log(\frac{M}{B})}\right)$
- Thus the disk striping technique is no good for sorting. To achieve optimal I/O bound we have to forget disk striping and use the disks independently.

Distribution sort

- In distribution sort we use a recursive process in which we use a set of S - 1 elements to partition the data into S disjoint buckets.
- A bucket contains items that are smaller than the ones in the next one.
- Equal buckets = best performance, it reduces the bucket size by a factor of $\Theta(S)$. Thus we only have $O(log_S n)$ levels of recursion.
- The challenge in distribution sort is to write the blocks of the buckets to the disk in an online manner and achieve a global balance by the end of the partitioning. So that the bucket can be read efficiently in the next recursion level.

Implementations

Vitter and Shriver developed two randomized online techniques for the partitioning so that with high probability each bucket is balanced across the D disks. In addition they use a partial striping in order to fit the pointers used to keep the layout of the buckets on the disk.

Partial striping is exactly what it says : it strips data across a subset of disks, clusters of size C. Here choosing C =√D does not change the optimality of sorting.

Computational geometry

- Here we deal with a large amount of computational geometry. These can be images, videos, simulations etc. One example is NASA's Earth Observation System project which produces petabytes of raster data per year. Data that needs to be further examined.
- The challenge is to find an efficient method to compute the data. Here for systems of this we need fast EM algorithms and data structures.
- Luckily, many problems on geometric objects can be reduced to a small core of problems, such as computing intersections, convex hulls, or nearest neighbors.

Basic problems

- Here we give some examples of basic problems used in solving geometry problems.
  - Computing the pairwise intersections of N segments in the plane
  - Finding all intersections between N non-intersecting red line segments and N non-intersecting blue line segments in the plane
  - Constructing the 2-D and 3-D convex hull of N points
  - Voronoi diagram and triangulation of N points in the plane

Computational geometry: EM algorithms/data structures

- Persistent B-trees, an offline method for constructing an optimal-space persistent version of the B-tree data structure, yielding a factor of B improvement over the generic persistence techniques.
- Batched Ltering, a general method for performing simultaneous EM searches in data structures that can be modeled as planar layered directed acyclic graphs; it is useful for 3-D convex hulls and batched point location. Multi-search on parallel computers is also considered here.
- Distribution sweeping, a parallel form of the sweeping algorithm implementend with the distribution paradigm saw earlier.