BBM371- Data Management

Lecture 11 External Sorting (multi-disk, multi-core)

27.12.2018

The Memory Hierarchy

- Registers, work at CPU-speed, a few hundreds of Bytes
- Different levels of cache memory, operating at 2 to 50 times CPU-speed, and ranging in size to a few Mbytes
- Main memory, operating at a few hundred times CPU-speed and comprising Gigabytes
- Hard disk or solid state disk, where access time is millions of cycles and size is several TBytes.
  - *The economical way to use disks is to transport data in large chunks*
- An analogous statement is true for any two adjacent levels of the hierarchy. The chunk size should be chosen such that the time for transferring a chunk is approximately equal to the time accessing a chunk.

The Parallel Disk Model of Aggarwal/Vitter

- It is usually phrased in terms of disks, but applies to any two adjacent levels of the memory hierarchy.
  - The machine has a CPU and a main memory of size M.
  - Data between main memory and disks is transferred in blocks of size B.
  - The machine has D disks that can be used in parallel.
- In one I/O-operation, one block of size B can be transferred between main memory and each disk.
- Algorithms are analyzed in terms of number of I/O-operations.

STXXL: Standard Template Library for Extra Large Data Sets

- The core of STXXL is an implementation of the C++ standard template library STL for external memory (out-of-core) computations
  - support of parallel disks
  - benefit from overlapping of I/O and computation
  - the I/O complexity of the algorithms remains optimal in most of the cases
  - asynchronous execution of the algorithmic components, enabling high-level task parallelism
- For internal computation, parallel algorithms from the MCSTL or the libstdc++ parallel mode are optionally utilized, making the algorithms inherently benefit from multi-core parallelism.

Merge Sort

- For N items to be sorted on a buffer of size B each:

  - Merge Sort on a single disk: $2\frac{N}{B}\left(1+\left\lceil log_{M/B}\frac{N}{M}\right\rceil\right)$=$2\frac{N}{B}\left\lceil log_{M/B}\frac{N}{B}\right\rceil$ disk accesses
  - Merge Sort on D parallel disks: $2\frac{N}{DB}\left\lceil log_{M/(DB)}\frac{N}{DB}\right\rceil$ disk accesses
  - This looks like the internal sorting.
  - Number of blocks is n=N/B. Instead of binary log, we have the logarithm to the memory size measured in number of blocks.

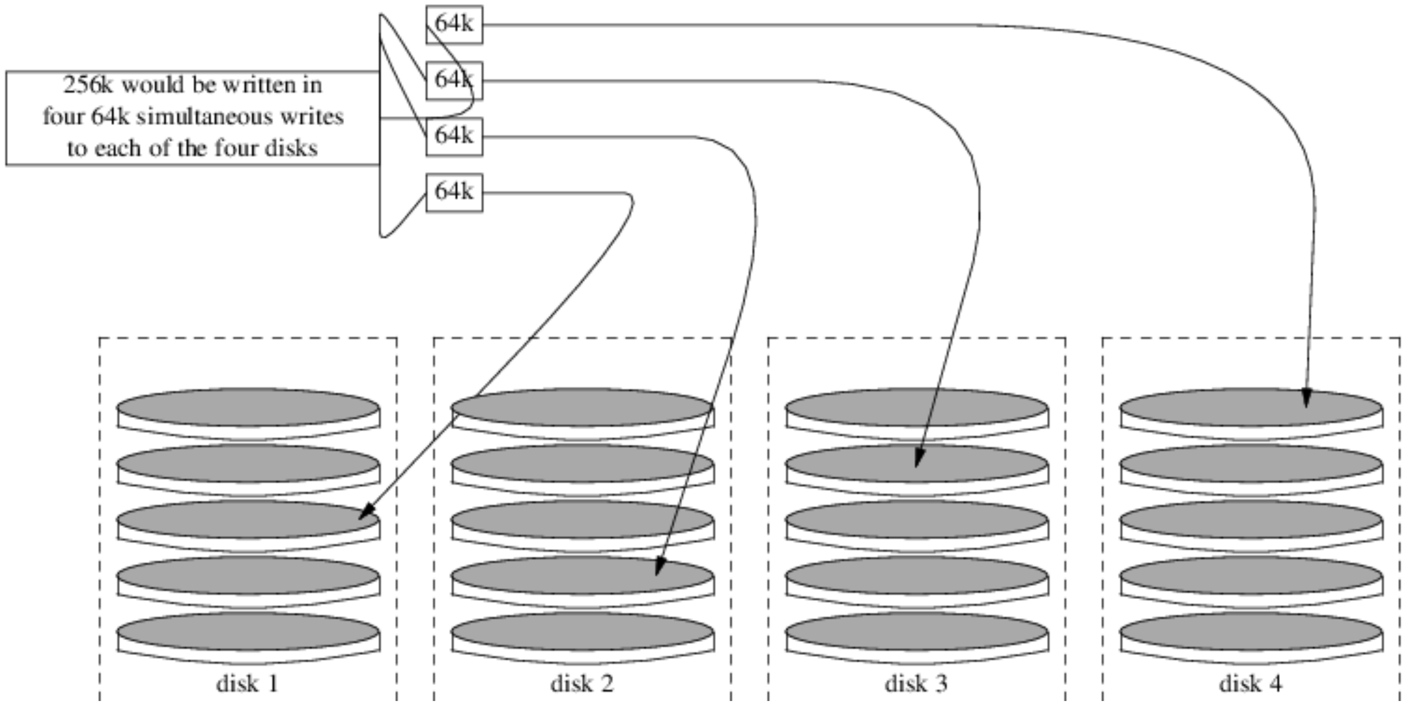
  - How good is this bound?
    - Merge Sort is optimal for one disk, but suboptimal for many disks.

Disk Striping

- We treat the D disks as a single disk with block size DB. A super-block of size DB consists of D blocks of size B.
- When a super-block is to be transferred, we transfer one standard block to each disk.
- We can generalize all single-disk results to D disks.
  - There might be more effective ways of using the D disks and that main memory can only hold $M/(DB)$ super-blocks.

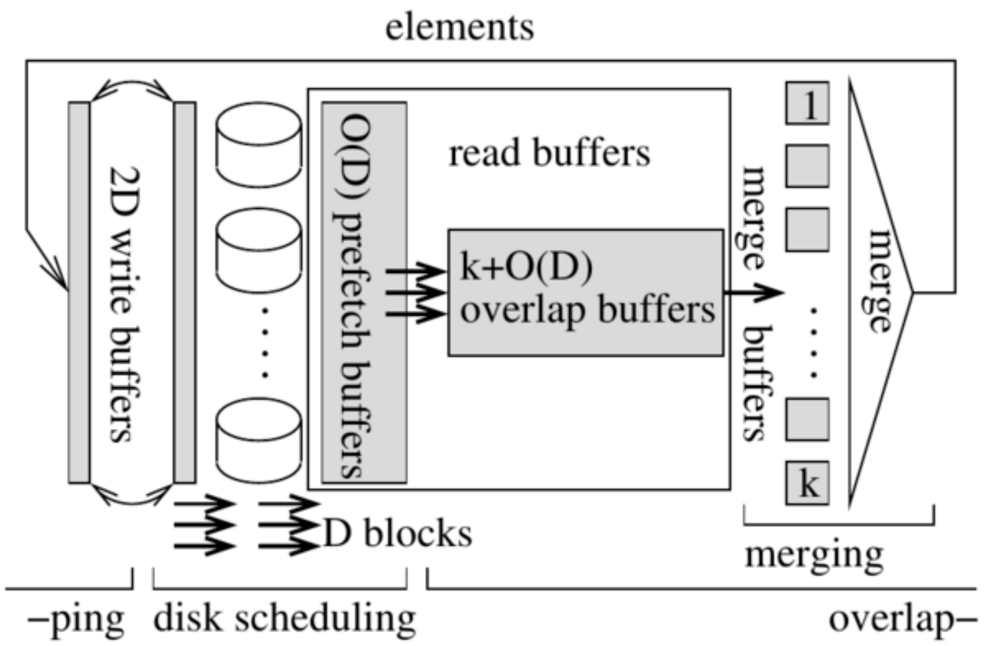
Disk Striping

- Treat D disks as a single disk with block size DB.
- A super-block of size DB consists of D blocks of size B.



Parallel Disk Sorting

- Parallel Disk Sorting
  - has an optimal I/O volume $\mathcal{O}(\frac{N}{DB}\log_{M/B}\frac{N}{B})$ (that matches te lower bound), and guarantees almost perfect overlap between I/O and computation.
  - https://stxxl.org/tags/1.4.1/design_algo_sorting.html



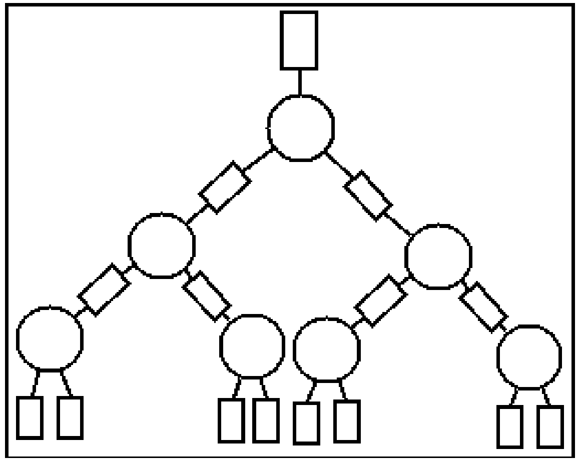
Cache-Oblivious Algorithms

- *Tall Cache Assumption*
  - Main memory consists of M/B blocks of size B.
  - In the case of caches, M/B is called the height of the cache and B is called the width of the cache.
  - The tall cache assumption states that the height is larger than the width, i.e., $M/B \geq B$. In other words, $M \geq B^2$
  - Many results about cache-oblivious algorithms hold true under the weaker assumption that $M \geq B^{1+\gamma}$ for some constant $\gamma > 0$.

- Cache-oblivious algorithms cannot use M and B in the program code. Nevertheless, they work well under the tall cache assumption.

Funnel Sort

- Funnel sort is a variant of merge sort.
- Split the input into smaller groups ●
  - Split N elements into $N^{(1/d)}$ groups of size $N^{(1-1/d)}$
  - We sort each part recursively.
  - We merge the sorted using *funnel merge (k-way merger)*
    - The memory layout is as in van-Emde-Boas trees..
- While being cache oblivious
  - $\Theta(n \log n)$ work
  - $\Theta((n \log n) / B)$ I/Os



Multi-Core Algorithms

- A multi-core is a parallel machine on a single chip.
  - There are several cores (= CPUs) on a single chip; up to 16 or 32 in commercial machines and up to 100 in experimental machines.
  - Each core has its own cache.
  - They share main memory
- Parallel External Memory (PEM)
  - We have P CPUs each with a private (fast) cache of size M.
  - The processors share a main memory; the main memory is unbounded in size and much slower than the private cache memories.
  - The private caches are partitioned into M/B blocks of size B each. Data is transferred in blocks between private caches and shared memory
  - In an I/O-step P blocks, one for each processor, can be transferred between main memory and private caches.
  - Concurrent read is supported. Concurrent write may or may not be supported.

Sorting

- We want to sort N elements on a PEM with P processors, each having a cache of size M. The block size is B.
  - The lower bound argument for sorting in external memory still works

$$\Omega\left(\frac{N}{PB}\log_{M/B} N/B\right)$$