Advanced Techniques for Combinatorial Algorithms: External-Memory Algorithms

Gianluca Della Vedova

Univ. Milano–Bicocca
https://gianluca.dellavedova.org

April 28, 2020

Memory Hierarchy

- CPU Registers
- L1 Cache: $32 - 256$ KBytes, latency $< 10^{-9}$ secs
- L2 Cache: $1 - 16$ MBytes, block transfer size $B = 32$ Bytes
- RAM: $1 - 32$ GBytes, $B = 64$ Bytes
- Disk: 100 GBytes - 1 PBytes, $B = 8$ KBytes, latency $> 10^{-3}$ secs

PDM model

- Parallel Disk Model
- Locality of reference
- Parallel disk access
- Disk striping (data across several disks)
- Count I/O operations

PDM parameters

- N = problem size (in units of data items)
- M = internal memory size (in units of data items)
- B = block transfer size (in units of data items)
- D = number of independent disk drives;
- P = number of CPUs
- Q = number of queries (for a batched problem);
- Z = answer size (in units of data items).
- $M < N$ and $1 \leq DB \leq M/2$
- $n = N/B$, $m = M/B$, $q = Q/B$, $z = Z/B$

Basic operations

- Scan: $\Theta(\frac{N}{DB}) = \Theta(\frac{n}{B})$
- Sort: $\Theta(\frac{N}{DB} \log_{M/B} \frac{N}{B}) = \Theta(\frac{n}{D} \log_{M/B} n)$
- Search: $\Theta(\log_{DB} N)$
- Output: $\Theta(\max\{1, \frac{Z}{DB}\}) = \Theta(\max\{1, \frac{z}{D}\})$

Disk striping

- I/O only on entire stripes
- cohesive set of disks
- D disks as a logical disk with logical block size DB

Main idea

- 1 disk: each I/O step transmits one block of size DB
- D disks: each I/O step consists of D simultaneous block transfers of size B each.
- Same number of I/O steps

Distribution sort

S buckets

- By choosing $S - 1$ pivots
- needs buckets of similar size, so $O(\log_S n)$ recursion layers
- scan to build the buckets. When a buffer is full $\Rightarrow$ write it
- $O(m)$ buckets
- probabilistic approach to select the pivots

Multiway Partitioning (PDM)

Multiway Partitioning

- $M = \{m_1, \ldots, m_d\}$ ordered set of pivots
- S: unordered set of elements
- A_i: i-th bucket. $a_i \in A_i$, $m_{i-1} < a_i \leq m_i$
- Goal: Compute A_is
- Goal: Compute $|A_i|$

Multiway Partitioning (PDM)

Algorithm 1: MultiPartition

```
Split A into sets S_1, ..., S_P;
foreach processor i in parallel do
    Read the vector of pivots M into the cache;
    Partition S_i into d buckets, J_i = number of items in each bucket
Prefix Sums on {J_1, ..., J_P} in parallel;
foreach processor i in parallel do
    Write elements S_i into memory locations offset appropriately by J_{i-1} and J_i
compute |A_i|s, using the prefix sums stored in J_P
```

PDM references

- Jeffrey Scott Vitter. Algorithms and Data Structures for External Memory. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2008 http://www.ittc.ku.edu/~jsv/Papers/Vit.IO_book.pdf
- L. Arge, M. T. Goodrich, M. Nelson, and N. Sitchinava. Fundamental parallel algorithms for private-cache chip multiprocessors. In Proceedings of SPAA '08, pages 197–206. ACM, 2008.
- Fundamental parallel algorithms for private-cache chip multiprocessors https://dl.acm.org/citation.cfm?id=1378573