Cache-Oblivious Priority Queue and Graph Algorithm Applications

Alexandru Furculita

Summary

I/O Model or external-memory model

Memory architecture:

- Internal memory of size M and,
- Arbitrarily large external memory partitioned into blocks of size B

Efficiency measure: the number of *memory transfers*, the number of blocks transfered between the two levels of memory

Limitations:

- Parameters B and M must be known
- Does not handle multiple memory levels
- Does not handle dynamic M

Cache-oblivious model

- Design and analyze algorithms in the I/O model, but without having the size of the memory and of the blocks as explicit parameters
- Analyze in the I/O model for optimal off-line cache replacement strategy: If the main memory is full, the ideal block in main memory is elected for replacement based on the future characteristics of the algorithm

Advantages:

- Optimal on arbitrary level optimal on all levels
- Portability: **B** and **M** not hard-wired into algorithm
- Dynamic changing parameters

Priority queues

- Maintains a set of elements each with a priority (or key) under the operations insert, delete, and deletemin
- Analyze in the I/O model for optimal off-line cache replacement strategy: If the main memory is full, the ideal block in main memory is elected for replacement based on the future characteristics of the algorithm

Priority queues in I/O Model

Problem

Sort $\boldsymbol{N}$ elements using an $O(\log_B N)$ priority queue closer to optimal solution.

The normal algorithms are a factor of $\frac{\log_B N}{\log_{\frac{M}{B}} \frac{N}{B}}$ from optimal

I/O efficient graph algorithms

Problem

Develop I/O-efficient algorithms for:

- list ranking
- Euler Tour
- BFS
- DFS

Summary

Optimal Cache-Oblivious Priority queues

Levels

- consists of $\Theta(\log \log N)$ levels of size N to a constant c
- the size of a level corresponds to the number of elements that can be stored within it

Optimal Cache-Oblivious Priority queues

Buffers

A level consists of two levels:

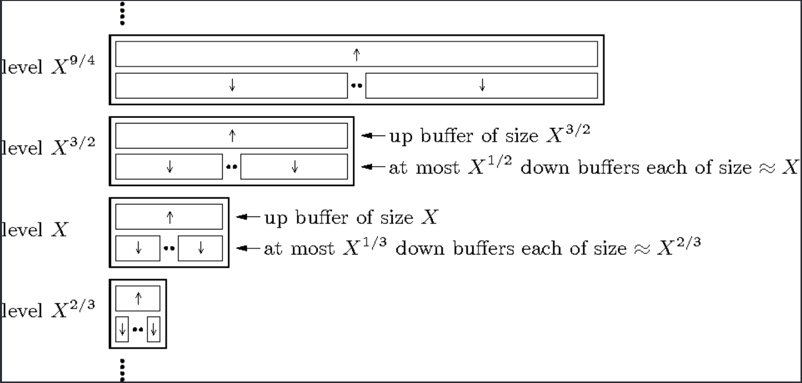
- *up buffer*: only one, can store up to X elements
- *down buffers:* at most $X^{\frac{1}{3}}$ such buffers, each containing between $\frac{1}{2}X^{\frac{2}{3}}$ and $2X^{\frac{2}{3}}$

Three invariants about the relationships between the elements if buffers of various levels are maintained:

- At level X, elements are sorted among the down buffers
- At level X, the elements in the down buffers have smaller keys than the elements in the up buffer
- The elements in the down buffers at level X have smaller keys than the elements in the down buffers at the next higher level

Optimal Cache-Oblivious Priority queues

Layout



Operations

Push

- Inserts X elements into level $X^{\frac{3}{2}}$
- Can be performed in $O(X^{\frac{1}{2}} + \frac{X}{B}\log_{\frac{M}{B}}\frac{X}{B})$ memory transfers amortized

Operations

Pull

- Removes the X elements with smallest keys from level $X^{\frac{3}{2}}$ and returns them in sorted order
- Can be performed in $O(1 + \frac{X}{B} \log_{\frac{M}{B}} \frac{X}{B})$ memory transfers amortized

Operations

Total cost

A set of N elements can be maintained in a linear-space cache-oblivious priority queue data structure supporting each insert, deletemin, and delete operation in $O(\frac{1}{B}\log_{\frac{M}{B}}\frac{N}{B})$ amortized memory transfers and $O(\log_2 N)$ amortized computation time.

Graph algorithms applications and results

Using the cache oblivious priority queue, there can be developed cache oblivious algorithms for several graph problems that uses the same number of memory accesses as a cache-aware algorithm:

- The list ranking, the Euler Tour, BFS, DFS, and centroid decomposition problems on a V node list can be solved in $O(sort(V))$ memory accesses
- The DFS or BFS tree of a directed graph can be computed in $O((V + \frac{E}{B})\log_2 V + sort(E)$ memory accesses.
- The BFS tree of an undirected graph can be computed cache-obliviously in $O(V + sort(E)$ memory accesses.

Bibliography

[1] Lars Arge, Michael A. Bender, Erik D. Demaine, Bryan Holland-Minkley, and J. Ian Munro. *An Optimal Cache-Oblivious Priority Queue and Its Application to Graph Algorithms*. Available at `http://epubs.siam.org/doi/abs/10.1137/S0097539703428324`.