External Memory

- Computationals models
- Shortest path in implicit grid graphs
  - RAM algorithm
  - I/O algorithms
  - Cache-oblivious algorithm

Philip Bille

Computational Models

- iPad Air 2.
  - A8X Chip (triple-core ARMv8-A)
  - L1 cache: 64 KB instruction + 64 KB data per core
  - L2 cache: 2 MB
  - L3 cache: 4 MB
  - Memory: 2 GB
  - Disk: 16 GB SSD

- Word RAM model.
  - Infinite memory of cells.
  - Read/write a cell.
  - Arithmetic and boolean operations (+,-,=,<,>,&,...)
- Cost.
  - Time complexity = number of operations.
Computational Models

- I/O model [Aggarwal and Vitter 1988].
  - Limited memory + infinite disk
  - I/O operation = read/write consecutive block of B cells between memory and disk.
  - Arithmetic and boolean operations (+,-,/,=,<,>,&,|) on cells in memory.
- Cost.
  - I/Os = number of I/O operations.
  - Computation is free (!)

Cache-oblivious model [Frigo et al. 1999].
- Identical to I/O model except algorithms do not know M and B.
- Program in RAM model and analyze in I/O model.
- Assume optimal cache replacement strategy with full associativity.
- Properties.
  - Efficient on one level of cache ⇒ efficient on all levels cache.
  - Portable + self-tunable + simple.

External Memory

- Computational models
- Shortest path in implicit grid graphs
  - RAM algorithm
  - I/O algorithms
  - Cache-oblivious algorithm

Shortest Paths in Implicit Grid Graphs

- Implicit grid graphs.
  - Let S and T be strings of length n.
  - The implicit grid graph for S and T is a 2D grid of (n+1) x (n+1) nodes.
  - For each node an edge to neighbors to E, S, SE.
  - E and S edges have weight 1.
  - SE edge (i-1,j-1) to (i, j) has weight 0 if S[i] = T[j] and 1 otherwise.
Shortest Paths in Implicit Grid Graphs

• Shortest paths in implicit grid graphs (SPIIG) problem.
  • Input. Strings S and T of length n.
  • Output. Length of shortest path from (0,0) to (n,n).

Shortest Paths in Implicit Grid Graphs

• Applications.
  • Shortest paths in implicit grid graphs is the edit distance problem.
  • With other edge weight functions we get longest common subsequence, sequence alignment, string similarity, approximate string matching, etc.

External Memory

• Computationals models
• Shortest path in implicit grid graphs
  • RAM algorithm
  • I/O algorithms
  • Cache-oblivious algorithm

RAM Algorithm

• How can we solve SPIIGG on a RAM?
• Dynamic programming algorithm.
  • Construct (n+1) x (n+1) matrix.
  • Fill in each entry in O(1) time in left-to-right top-to-bottom order.
• Time. O(n²)
• Space. O(n) (only store current + last row)
• Slightly faster solutions known [MP1980, Myers1999, CLZ2002, BFC2008]
External Memory

• Computationals models
• Shortest path in implicit grid graphs
  • RAM algorithm
  • I/O algorithms
  • Cache-oblivious algorithm

External Memory Algorithms

• Goal. Efficient external memory algorithms.
  • I/O model.
    • Solution 1. Converted RAM algorithm
    • Solution 2. Table partitioning
  • Cache-oblivious model.
    • Solution 3. Recursive table partitioning

Solution 1. Converted RAM Algorithm

• Strings S and T stored consecutively in n/B blocks on disk.

  • Algorithm.
    • Do as RAM algorithm. Read and write blocks as necessary.
  • I/Os. O(n²/B).

Solution 2. Table Partitioning

• Divide into subtables with overlapping boundaries.

  • Algorithm. Process subtables from left-to-right, top-to-bottom order. For each subtable:
    • Read corresponding substrings and input boundary into internal memory
    • Fill in subtable using RAM algorithm.
    • Write output boundary to disk.
How to choose subtable size?
- Make subtable $dM \times dM$ for $d < 1$ such that substrings + input boundary + output boundary + space for internal memory algorithm on subtable $< M$.

Number of subtables = $O(n^2/M^2)$.

I/Os per subtable = $O(M/B)$.

$\Rightarrow O(n^2/M^2 \cdot M/B) = O(n^2/MB)$

Solution 2. Table Partitioning
- Theorem. We can solve SPIIGG in the I/O model in
  - $O(n^2/MB + n/B)$ I/Os
  - $O(n^2)$ time
  - $O(n)$ space

Solution 2. Table Partitioning
- Algorithm.
  - Divide table into 4 quadrants with overlapping boundaries.
  - Process quadrants from left-to-right, top-to-bottom order. For each quadrant:
    - Read corresponding substrings and input boundary.
    - Fill in quadrant recursively.
    - Write output boundary.

Solution 3. Recursive Table Partitioning
Solution 3. Recursive Table Partitioning

- I/Os.
- Define $IO(n)$ = number of I/Os to process a table of size $n \times n$
  - Case 1: $n \leq dM$ (substrings + boundaries + computation fit in internal mem)
    - $IO(n) = O(n/B)$
  - Case 2: $n > dM$?

Solution 3. Recursive Table Partitioning

- **Algorithm.**  
  - Divide table into 4 quadrants with overlapping boundaries. 
  - Process quadrants from left-to-right, top-to-bottom order. For each quadrant:
    - Read corresponding substrings and input boundary. 
    - Fill in quadrant recursively. 
    - Write output boundary.

Solution 3. Recursive Table Partitioning

- Case 1 + 2:

  $IO(n) = \begin{cases} 
  O(n/B) & \text{if } n \leq dM \\
  4 \cdot IO(n/2) + O(n/B) & \text{if } n > dM 
  \end{cases}$

  $\Rightarrow IO(n) = O(n^2/MB)$

Solution 3. Recursive Table Partitioning

- **Theorem.** We can solve SPIIGG in the cache-oblivious model in
  - $O(n^2/MB + n/B)$ I/Os
  - $O(n^2)$ time
  - $O(n)$ space
External Memory

- Computationals models
- Shortest path in implicit grid graphs
  - RAM algorithm
  - I/O algorithms
  - Cache-oblivious algorithm