Cache-Oblivious Search Trees via Trees of Small Height

Gerth Stølting Brodal
BRICS
University of Aarhus

Joint work with Rolf Fagerberg and Riko Jacob
Result: New Search Tree

\{ 1, 3, 4, 5, 6, 7, 8, 10, 11, 13 \}

\[
\begin{array}{cccccccc}
\end{array}
\]
Outline

• Models of Computation
  – I/O Model
  – Cache-Oblivious Model
• Cache-Oblivious Search Trees
  – Static
  – Dynamic
• Experiments
  – Memory Layouts of Trees
• Summary
I/O Model

- Bottleneck $\equiv$ I/Os between the two highest memory levels
- B-trees support searches and updates in $O(\log_B N)$ I/Os
- $\Theta \left( \frac{M}{B} \right)$-way merge-sort achieves optimal $\Theta \left( \frac{N}{B} \log_{M/B} \frac{N}{B} \right)$ I/Os

$N = \text{problem size}$

$M = \text{memory size}$

$B = \text{I/O block size}$

Aggarwal and Vitter 1988
Cache-Oblivious Model

- I/O model
- Algorithms do not know the parameters $B$ and $M$
- Optimal off-line cache replacement strategy

Examples
- Scanning, Linear time selection
- Matrix-transposition, FFT, Funnel-sorting

Lemma
Optimal cache-oblivious algorithm implies optimal algorithm on each level of a fully associative multi-level cache using LRU

Frigo et al. 1999
Outline

- Models of Computation
  - I/O Model
  - Cache-Oblivious Model
- Cache-Oblivious Search Trees
  - Static
  - Dynamic
- Experiments
  - Memory Layouts of Trees
- Summary
Static Cache-Oblivious Trees

Recursive memory layout ≡ van Emde Boas layout

Degree $O(1)$

Searches use $O\left(\log_B N\right)$ I/Os

Range reportings use $O\left(\log_B N + \frac{k}{B}\right)$ I/Os

Prokop 1999
Dynamic Cache-Oblivious Trees

Search: $O(\log_B N)$
Range Reporting: $O\left(\log_B N + \frac{k}{B}\right)$
Updates: $O\left(\log_B N + \frac{\log^2 N}{B}\right)$

- Pointer Based Strongly Weight Balanced B-trees
- Dynamic van Emde Boas Layout
- Packed Memory Management
- Two Levels of Indirection

Arge and Vitter 1996
Prokop 1999
Itai et al. 1981
Bender, Demain, Farach-Colton 2000
Binary Trees of Small Height

- If an insertion causes non-small height then rebuild subtree at nearest ancestor with sufficient few descendents
- Insertions require amortized time $O(\log^2 N)$

Andersson and Lai 1990
Dynamic Cache-Oblivious Trees

- Embed a dynamic tree of small height into a complete tree
- Static van Emde Boas layout

![Dynamic Cache-Oblivious Trees Diagram]

- **Search**: $O(\log_B N)$
- **Range Reporting**: $O\left(\log_B N + \frac{k}{B}\right)$
- **Updates**: $O\left(\log_B N + \frac{\log^2 N}{B}\right)$

New
Example
Outline

• Models of Computation
  – I/O Model
  – Cache-Oblivious Model
• Cache-Oblivious Search Trees
  – Static
  – Dynamic
  ▶ Experiments
    – Memory Layouts of Trees
• Summary
Memory Layouts of Trees

DFS

BFS

inorder

van Emde Boas
van Emde Boas layout wins, followed by the BFS layout
Searches with Implicit Layouts

- BFS layout wins due to simplicity and caching of topmost levels
- van Emde Boas layout requires quite complex index computations
Implicit vs Pointer Based Layouts

- Implicit layouts become competitive as $n$ grows.
Insertions in Implicit Layouts

- Insertions are rather slow (factor 10-100 over searches)
Summary

- New simple cache-oblivious search trees
  
  **Search** \(O(\log_B N)\)
  
  **Range Reporting** \(O\left(\log_B N + \frac{k}{B}\right)\)
  
  **Updates** \(O\left(\log_B N + \frac{\log^2 N}{B}\right)\)

- Update time \(O(\log_B N)\) by one level of indirection (implies sub-optimal range reporting)

- Importance of memory layouts

- van Emde Boas layout gives good cache performance

- Computation time is important when considering caches