On the Cell Probe Complexity of Dynamic Membership

or

Can We Batch Up Updates in External Memory?

Ke Yi and Qin Zhang

Hong Kong University of Science & Technology

SODA 2010
Jan. 17, 2010
The power of buffering

- For numerous dynamic data structure problems in external memory, updates can be buffered.
  - Buffer tree [Arge 1995]
  - Logarithmic method [Bentley 1980] + B-tree
The power of buffering

- For numerous dynamic data structure problems in external memory, updates can be buffered.
- Buffer tree [Arge 1995]
- Logarithmic method [Bentley 1980] + B-tree

<table>
<thead>
<tr>
<th>problem</th>
<th>update</th>
<th>query</th>
<th>cache-oblivious</th>
</tr>
</thead>
<tbody>
<tr>
<td>stack</td>
<td>$O(1/b)$</td>
<td>/</td>
<td>trivial</td>
</tr>
<tr>
<td>queue</td>
<td>$O(1/b)$</td>
<td>/</td>
<td>trivial</td>
</tr>
<tr>
<td>priority-queue</td>
<td>$O(\frac{1}{b} \log_b n)$</td>
<td>/</td>
<td>[Arge et. al. STOC 02]</td>
</tr>
<tr>
<td>predecessor</td>
<td>$O(\frac{1}{b} \log n)$</td>
<td>$O(\log n)$</td>
<td>trivial</td>
</tr>
<tr>
<td>range-sum</td>
<td>$O(\frac{b^e}{b} \log n)$</td>
<td>$O(\log_b n)$</td>
<td>[Brodal et. al. this SODA]</td>
</tr>
<tr>
<td>range-reporting...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$b$: size of a block/cell (in words)
How about Dictionary and Membership?

- Dictionary and membership (selected)
  - Knuth, 1973: **External hashing**
    Expected average cost of an operation is $1 + 1/2^{\Omega(b)}$, provided the load factor $\alpha$ is less than a constant smaller than 1. (truly random hash function)
  - Data structures like Arge’s **Buffer tree**:
    \[\text{Update} = O\left(\frac{b^\epsilon}{b} \log n\right), \text{Query} = O(\log_b n).\]
How about Dictionary and Membership?

- Dictionary and membership (selected)
  - Knuth, 1973: External hashing
    Expected average cost of an operation is $1 + 1/2^{\Omega(b)}$, provided the load factor $\alpha$ is less than a constant smaller than 1. (truly random hash function)
  - Data structures like Arge’s Buffer tree:
    Update = $O(\frac{b^e}{b} \log n)$, Query = $O(\log_b n)$.

- Question: can we improve the amortized update cost to $o(1)$ in external memory, without sacrificing the query speed by much?
The conjecture

A *long-time folklore* conjecture in external memory community: (explicitly stated by Jensen and Pagh, 2007)

\[ t_u \text{ must be } \Omega(1) \text{ if } t_q \text{ is required to be } O(1) \]

\[ t_u: \text{ expected amortized update cost} \]
\[ t_q: \text{ expected average query cost} \]
The conjecture

A long-time folklore conjecture in external memory community: (explicitly stated by Jensen and Pagh, 2007)

\[ t_u \text{ must be } \Omega(1) \text{ if } t_q \text{ is required to be } O(1) \]

Our small step: \( \leq 1.1 \)

\[ t_u: \text{ expected amortized update cost} \]
\[ t_q: \text{ expected average query cost} \]
Problems

Membership: Maintain a set $S \subseteq U$ with $|S| \leq n$. Given an $x \in U$, is $x \in S$? Yes or No.

Dictionary: If $x \in S$, return associated info, otherwise say No. Often assumes “indivisibility”.

Objective: Tradeoff between update cost $t_u$ and query cost $t_q$

Two of the most fundamental data structure problems in computer science!
The computational model

- The cell probe model [Yao 1981] with a content preserving cache
- A data structure is a collection of $b$-bit cells
- Cost of an operation: \# of cells read/changed
- A cache of $m$-bits; probing the cache is free
The computational model

- The cell probe model [Yao 1981] with a content preserving cache
  - A data structure is a collection of $b$-bit cells
  - Cost of an operation: number of cells read/changed
  - A cache of $m$-bits; probing the cache is free
- This is essentially the external memory model.
The computational model

- The cell probe model [Yao 1981] with a content preserving cache
- A data structure is a collection of $b$-bit cells
- Cost of an operation: # of cells read/changed
- A cache of $m$-bits; probing the cache is free
- This is essentially the external memory model.

The cell size $b$ ranges from 1 to $\log u$ up to $n^\epsilon$. 

Our results hold for arbitrary $b$, though they are more meaningful for large $b$'s.
The computational model

- The cell probe model [Yao 1981] with a content preserving cache
- A data structure is a collection of $b$-bit cells
- Cost of an operation: $\#$ of cells read/changed
- A cache of $m$-bits; probing the cache is free
- This is essentially the external memory model.
- The cell size $b$ ranges from 1 to $\log u$ up to $n^\epsilon$.

  Our results hold for arbitrary $b$, though they are more meaningful for large $b$’s

- The cache may not affect $t_q$ by much, but does affect $t_u$ in almost all common data structures (typically $o(1)$).
Let’s go!

**Membership**

**Problem:** Maintain a set $S \subseteq U$.
Given $x \in U$, is $x \in S$?

**Goal:** tradeoff between $t_u$ and $t_q$

Membership $t_q = 1 + \delta$
($0 \leq \delta < 1/2$) [this paper]

Without indivisibility assumption

Dictionary (successful)
[SPAA 09] Wei, Yi and Zhang
Outline

- A model for queries
Outline

- A model for queries
- Deterministic algorithm + random update sequence
Outline

- A model for queries
- **Deterministic** algorithm + random update sequence

Can be extended to:

- randomized algorithm
- the case with **error**
Outline

- A model for queries
- **Deterministic algorithm + random update sequence**
  
  Can be extended to:
  
  - randomized algorithm
  - the case with error

- Future work
Preliminaries

- $U = \{0, 1, \ldots, u - 1\}$: universe. $|U| = u$.

- $m$: size of cache. In bits.
  - $b$: size of one cell. In bits.

- $n$: total number of inserted elements.

- $S$: set of elements we are maintaining. $|S| \leq n$
Preliminaries

- \( U = \{0, 1, \ldots, u - 1\} \): universe. \(|U| = u\).

- \( m \): size of cache. In \text{bits}.

- \( b \): size of one cell. In \text{bits}.

- \( n \): total number of inserted elements.

- \( S \): set of elements we are maintaining. \(|S| \leq n\)

- A very mild assumption
  - \( u \geq \Omega(n) \geq \Omega(mb)\)
The model

\[
\psi_M(x) = \begin{cases} 
1, & x \in S; \\
0, & x \notin S. 
\end{cases}
\]

query cost: 0

\[
\pi_M(x) = 0
\]

otherwise

\[
B_{\pi_M(x)}
\]

content of cell

\[
\pi_M(x)
\]

disk

cell selector

cache

Query \(x\)

M

\[\text{query cost: 0}\]
The model

\[ \psi_M(x) = \begin{cases} 
1, & x \in S; \\
0, & x \notin S.
\end{cases} \]

\[ \pi_M(x) = 0 \]

otherwise

\[ f_{M,B\pi_M(x)}(x) = \begin{cases} 
1, & x \in S; \\
0, & x \notin S; \\
\ast, & \text{unknown.}
\end{cases} \]

query cost: 0

query cost: 1

query cost: \geq 2

disk

M

cache

\[ B_{\pi_M(x)} \]
The model

\[ D(x) = \begin{cases} 
\psi_M(x), & \text{if } \pi_M(x) = 0; \\
\hat{f}_{M,B\pi_M(x)}(x), & \text{otherwise.} 
\end{cases} \]
The model

\[ \mathcal{D}(x) = \begin{cases} 
\psi_M(x), & \text{if } \pi_M(x) = 0; \\
 f_{M,B_{\pi_M(x)}}(x), & \text{otherwise.}
\end{cases} \]

Families of functions \{\pi\}, \{\psi\}, \{f\} are fixed
The model

During an update

pre $\begin{array}{|c|c|c|c|}
M & B_1 & B_2 & B_3 \\
\hline
\end{array}$

post $\begin{array}{|c|c|c|c|}
M' & B_1' & B_2' & B_3' \\
\hline
\end{array}$

Query $x$

$\psi_M(x) = 0$

otherwise

$\pi_M(x) = 0$

$B_{\pi_M(x)}$

$\mathcal{D}(x) = \begin{cases} 
\psi_M(x), & \text{if } \pi_M(x) = 0; \\
\psi_M(x), & \text{otherwise.}
\end{cases}$
The model

Query $x$

$\psi_M(x)$

$\pi_M(x) = 0$

otherwise

$\mathcal{D}(x) = \begin{cases} 
\psi_M(x), & \text{if } \pi_M(x) = 0; \\
\mathcal{f}_{M,B\pi_M(x)}(x), & \text{otherwise.}
\end{cases}$

During an update

pre

\begin{array}{c|c|c|c|c|}
M & B_1 & B_2 & B_3 & \cdots & B_d \\
\downarrow & \text{free} & \downarrow & \text{cost 1 if } B_j \neq B'_j
\end{array}

post

\begin{array}{c|c|c|c|c|}
M' & B'_1 & B'_2 & B'_3 & \cdots & B'_d \\
\end{array}
Framework of the proof

- During the insertion sequence,
  1. neglect first $\sigma n$ elements,
  2. divide the rest into rounds; each contains $s$ elements.

We focus on (implicit) queries at the end of each round.
Framework of the proof

- During the insertion sequence,
  1. neglect first $\sigma n$ elements,
  2. divide the rest into rounds; each contains $s$ elements.

We focus on (implicit) queries at the end of each round.

- Let $B_i^{\text{pre}}$ and $B_i^{\text{post}}$ be the states of cell $i$ at the beginning and the end of a round $R$.

- Cells $i$ having $B_i^{\text{pre}} \neq B_i^{\text{post}}$ must be modified in round $R$. 
Framework of the proof

- During the insertion sequence,
  1. **neglect** first $\sigma n$ elements,
  2. **divide** the rest into **rounds**; each contains $s$ elements.

We focus on (implicit) queries at the end of each round.

- Let $B^\text{pre}_i$ and $B^\text{post}_i$ be the states of cell $i$ at the beginning and the end of a round $R$.

- Cells $i$ having $B^\text{pre}_i \neq B^\text{post}_i$ must be modified in round $R$.

We try to show that during each round:

- at least $\Omega(s)$ cells $i$ have $B^\text{pre}_i \neq B^\text{post}_i$.

  $\rightarrow$ amortized update cost is $\Omega(1)$
High level ideas of the proof (focus on 1 round)

Consider queries at the final snapshot of a round.

1. The cache alone cannot answer too many queries.
   Intuition: $2^m \ll \binom{u}{\epsilon \cdot n}$
High level ideas of the proof (focus on 1 round)

Consider queries at the final snapshot of a round.

1. The cache alone cannot answer too many queries.
   Intuition: \[2^m \ll \left(\frac{u}{e \cdot n}\right)^2\]

For a fixed cache state \(M\)

2. At any time \(\geq \Omega(n)\) insertions, \# of \(x\) "\(D(x) = *\)“ is small.
   Reason: by the constraint \(t_q \leq 1.1\)
   \(\text{answer is unknown after 1 disk probe}\)
High level ideas of the proof (focus on 1 round)

Consider queries at the final snapshot of a round.

1. The cache alone cannot answer too many queries.
   Intuition: $2^m \ll \left(\frac{u}{\epsilon \cdot n}\right)$

For a fixed cache state $M$

2. At any time $\geq \Omega(n)$ insertions, # of $x$ “$D(x) = *$” is small.
   Reason: by the constraint $t_q \leq 1.1$

3 (because of 2). Cell selector $\pi(\cdot)$ used has to be balanced.
   Intuition: otherwise the data structure will not be correct, under a random insertion sequence w.h.p.

Let $\alpha_i = |\{x | \pi(x) = i\}| / u$. $\pi(\cdot)$ is balanced if there are not too many $\alpha_i \geq \Omega\left(\frac{b}{n}\right)$
High level ideas of the proof (focus on 1 round)

Consider queries at the final snapshot of a round.

1. The cache alone cannot answer too many queries.
   Intuition: \(2^m \ll \binom{u}{\epsilon \cdot n}\)

For a fixed cache state \(M\)

2. At any time \(\geq \Omega(n)\) insertions, \# of \(x\) \(\mathcal{D}(x) = *\) is small.
   Reason: by the constraint \(t_q \leq 1.1\)

3 (because of 2). Cell selector \(\pi(\cdot)\) used has to be balanced.
   Intuition: otherwise the data structure will not be correct,
   under a random insertion sequence w.h.p.

1 + 3 \(\Rightarrow\) 4. In a round, inserted elements’ query paths go to
many different cells after probing the cache.
High level ideas of the proof (cont.)

5. $\Omega(s)$ cells have to change.

Intuition: new elements are chosen randomly from $U$. For cell $i$, no matter what $B_{i}^{\text{pre}}$ is, if \{f_{M,B_{i}^{\text{post}}} (x) \mid \pi_{M}(x) = i\} contains few “∗”, then $B_{i}^{\text{pre}} \neq B_{i}^{\text{post}}$ with high probability.
High level ideas of the proof (cont.)

5. $\Omega(s)$ cells have to change.

Intuition: new elements are chosen randomly from $U$. For cell $i$, no matter what $B^\text{pre}_i$ is, if $\{f_{M,B^\text{post}_i}(x) \mid \pi_M(x) = i\}$ contains few "∗", then $B^\text{pre}_i \neq B^\text{post}_i$ with high probability.

Finally,

- (2) – (5) hold with high probability $\left(1 - e^{-\Omega(n)}\right)$, therefore hold for all $2^m$ states of $M$ w.h.p.
- Total cost per round is $\Omega(s)$
- Amortized cost per insertion is at least $\Omega(s) \cdot (1 - \sigma)n/s \cdot 1/n \geq \Omega(1)$. 
High level ideas of the proof (cont.)

5. \( \Omega(s) \) cells have to change.

Intuition: new elements are chosen randomly from \( U \). For cell \( i \), no matter what \( B_i^{\text{pre}} \) is, if \( \{ f_{M,B_i^{\text{post}}} (x) \mid \pi_M(x) = i \} \) contains few "*", then \( B_i^{\text{pre}} \neq B_i^{\text{post}} \) with high probability.

Finally,

- (2) – (5) hold with high probability \( (1 - e^{-\Omega(n)}) \), therefore hold for all \( 2^m \) states of \( M \) w.h.p.
- Total cost per round is \( \Omega(s) \)
- Amortized cost per insertion is at least \( \Omega(s) \cdot (1 - \sigma)n/s \cdot 1/n \geq \Omega(1) \).
Latest results

\[ t_q = 1 + \delta \]

General Hashing (successful)
assume indivisibility

General Membership

(0 < \delta < 1)
Latest results

Very recently with Elad Verbin, we proved this conjecture (even more): If $t_u \leq 0.99$, then $t_q$ is required to be $\Omega(\log b \log n \frac{n}{m})$.

- A strong dichotomy result:
  
  Hash or Buffer-tree!

- Completely different techniques

$14-2$
Further work

- We still cannot handle fast updates.
  
e.g. if $t_u = O(1/b)$, $t_q = \Omega(n^\epsilon)$?
Further work

- We still cannot handle fast updates.
  e.g. if $t_u = O(1/b)$, $t_q = \Omega(n^\epsilon)$?

- Lower bounds of other dynamic problems in the external memory.
  e.g., for union-find, need super-log query time
    if we want to batch up the updates?
  Call for new techniques?
Further work

- We still cannot handle fast updates.  
  e.g. if $t_u = O(1/b)$, $t_q = \Omega(n^\epsilon)$?

- Lower bounds of other **dynamic problems** in the external memory.  
  e.g., for union-find, need **super-log query time**  
  if we want to batch up the updates?  
  Call for new techniques?

- Can we **simplify** the complicated **combinatorial** proof?  
  Use, e.g., **encoding arguments** like Pătraşcu-Viola.
The End

THANK YOU

Q and A