Advanced Data Structures

Lecture 07: Packed and Compressed Hash Tables

Florian Kurpicz
New Topic: External Memory Hash Tables

- now hash tables
- first packed and compressed hash table
- presented in January ’23 at ALENEX
Motivation

Setting
- static hash table for objects of variable size
- storing objects in external memory
- ideally retrieve objects in single I/O
- very small internal memory data structure

Objects of Variable Size
- only blocks of size $B$ bits can be transferred
- one I/O per block transfer

External Memory
Space-Efficient Object Stores from Literature

<table>
<thead>
<tr>
<th>Method</th>
<th>$I_b$</th>
<th>load factor</th>
<th>I/Os</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fixed</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Larson et al. [LR85]</td>
<td>96</td>
<td>&lt;96 %</td>
<td>1</td>
</tr>
<tr>
<td>SILT SortedStore [Lim+11]</td>
<td>51</td>
<td>100 %</td>
<td>1</td>
</tr>
<tr>
<td>Linear Separator [Lar88]</td>
<td>8</td>
<td>85 %</td>
<td>1</td>
</tr>
<tr>
<td>Separator [GL88; LK84]</td>
<td>6</td>
<td>98 %</td>
<td>1</td>
</tr>
<tr>
<td>Robin Hood [Cel88]</td>
<td>3</td>
<td>99 %</td>
<td>1.3</td>
</tr>
<tr>
<td>Ramakrishna et al. [RT89]</td>
<td>4</td>
<td>80 %</td>
<td>1</td>
</tr>
<tr>
<td>Jensen, Pagh [JP08]</td>
<td>0</td>
<td>80 %</td>
<td>1.25</td>
</tr>
<tr>
<td>Cuckoo [Aza+94; Pag03]</td>
<td>0</td>
<td>&lt;100 %</td>
<td>2</td>
</tr>
<tr>
<td>PaCHash, $a = 1$</td>
<td>2</td>
<td>100 %</td>
<td>2*</td>
</tr>
<tr>
<td>PaCHash, $a = 8$</td>
<td>5</td>
<td>100 %</td>
<td>1.13*</td>
</tr>
<tr>
<td><strong>variable</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SILT LogStore [Lim+11]</td>
<td>832</td>
<td>100 %</td>
<td>1</td>
</tr>
<tr>
<td>SkimpyStash [DSL11]</td>
<td>32</td>
<td>≤98 %</td>
<td>8</td>
</tr>
<tr>
<td>PaCHash, $a = 1$</td>
<td>2</td>
<td>99.95 %</td>
<td>2.06*</td>
</tr>
<tr>
<td>PaCHash, $a = 8$</td>
<td>5</td>
<td>99.95 %</td>
<td>1.19*</td>
</tr>
</tbody>
</table>
PaCHash Overview

- Objects of variable size
- Hash function $h: K \rightarrow 1..am$
- Sorted objects in EM in bins
- First bin (partially) in block
- No fragmentation
- Store offset in EM
- $\bar{b}$ bits remaining per EM block
- $p = \langle p_1, \ldots, p_m \rangle$
Finding Blocks

### Query Algorithm

- \( b_x = h(x) \)
- find first \( i \) with \( p_i \leq b_x \)
- if \( p_i = b_x \) let \( i = i - 1 \)
- find first \( j \) with \( p_j > b_x \)
- return \( i .. (j - 1) \)

### Elias-Fano Coding

- given \( k \) monotonic increasing integers in \( 1 .. u \)
- store \( \log k \) MSBs encoded in bit vector
- store \( \log(\frac{u}{k}) \) LSBs plain
- \( k(2 + \log(\frac{u}{k})) + 1 + o(k) \) bits in total
- predecessor in \( O(k) \) time

### Lemma: Space with Elias-Fano Coding

When using Elias-Fano coding [Eli74; Fan71] to store \( p \), the index needs \( 2 + \log a + o(1) \) bits of internal memory per block.
Lemma: Expected Predecessor Time

When using Elias-Fano coding to store $p$, the range of blocks containing the bin of an object $x$ can be found in expected constant time.

Proof (Sketch)

- consider $\lceil \log m \rceil$ MSB
- let bin $b_x$ have MSBs equal to $u$
- expected size $E(Y_u)$ of all bins with MSB $u$ that are $< b_x$ is

$$\sum_{y \in S} |y| \cdot \mathbb{P}(h(y) \text{ w/ MSB } = u; h(y) < h(x))$$

$$\leq \sum_{y \in S} |y| \cdot \mathbb{P}(h(y) \text{ w/ MSB } = u)$$

$$= \frac{1}{m} \sum_{y \in S} |y| = \frac{m\bar{B}}{m} = \bar{B}$$

- number of entries to scan is $E(Y_u)/\bar{B} = 1$
Lemma: Additional Blocks Loaded

Retrieving an object $x$ of size $|x|$ from a PaCHash data structure loads $\leq 1 + |x|/\bar{B} + 1/a$ consecutive blocks from the external memory in expectation.

Proof (Sketch)

- expected size of bin $b_x = h(x)$

$$\mathbb{E}(|b_x|) = |x| + \sum_{y \in S, y \neq x} |y| \mathbb{P}(y \in b_x)$$

$$\leq |x| + \sum_{y \in S} |y| \mathbb{P}(y \in b_x)$$

$$= |x| + \sum_{y \in S} |y| \cdot \frac{1}{am} = |x| + \frac{\bar{B}}{a}$$

Proof (Sketch, cnt.)

- expected number of blocks overlapped by $b_x$

$$\mathbb{E}(X) = 1 + \left(\mathbb{E}(|b_x|) - 1\right)/\bar{B}$$

$$= 1 + \frac{|x|}{\bar{B}} + \frac{1}{a} - 1/\bar{B}$$

- $\mathbb{P}$(bin and block border align) $= 1/\bar{B}$
## Hardware and Software
- Intel i7 11700 (base clock speed: 2.5 GHz)
- 1 TB Samsung 980 Pro NVMe SSD
- Ubuntu 21.10 (Kernel 5.13.0)
- `io_uring` for I/O operations
- GCC 11.2.0 (-O3 -march=native)
- $B = 4096$ bytes

## Objects
- here only **fixed size**
- more in the paper (very similar results)

## Competitors
- LevelDB [Goo21]
- RocksDB [Fac21]
- SILT [Lim+11]
- `std::unordered_map`
- RecSplit [EGV20]
- CHD [BBD09; CR+12]
- PTHash [PT21]
Construction

Number of objects [Millions]

Space internal [B/object]

\(10^{-1}\)

\(10^0\)

\(10^1\)

\(10^2\)

2 4

Number of objects [Millions]

Space external [B/object]

\(265\)

\(270\)

\(275\)

\(280\)

Number of objects [Millions]

Construction Throughput buffered I/O [Objects/s]

\(10^6\)

\(10^7\)

Number of objects [Millions]

---

- CHD (16-perfect) [BBD09]
- LevelDB [Goo21]
- RecSplit [EGV20]
- SILT (Static part) [Lim+11]
- Cuckoo (here)
- PTHash [PT21]
- RocksDB [Fac21]
- Separator (here)
- LevelDB (Static part) [Goo21]
- PaCHash (here)
- SILT [Lim+11]
- std::unordered_map
Queries

Number of objects [Millions]

Query Throughput

CHD (16-perfect) [BBD09]
LevelDB [Goo21]
RecSplit [EGV20]
SILT (Static part) [Lim+11]
Cuckoo (here)
PTHash [PT21]
RocksDB [Fac21]
Separator (here)
LevelDB (Static part) [Goo21]
PaCHash (here)
SILT [Lim+11]
std::unordered_map
Maximum Load Factor of Competitors

Cuckoo Hashing

Separator Hashing

Average object size [B]

Identical size  Normal distribution  Uniform distribution
Lemma: Space with Succincter

When using Succincter [Pat08] to store \( p \), the index needs 1.44 + \( \log(a + 1) \) + \( o(1) \) bits of internal memory per block.

Structure of Bit Vector
- runs of 0s and 10s
- sometimes additional 1s

Entropy Encoding
- encode positions directly
- compress bit vector using Huffman codes
- encode blocks of size 8, 16, 32, or 64

![Graph showing space and query throughput vs. parameter a for different data structures.]

- Huffman, Twitter
- Huffman, UniRef
- Huffman, Wikipedia
- Elias-Fano, Twitter
- Elias-Fano, UniRef
- Elias-Fano, Wikipedia
- Succincter (theoretical)
Conclusion and Outlook

This Lecture
- PaCHash

Next Lecture
- more on hashing
Bibliography I


Bibliography II


Bibliography III


