6.886 talk: Engineering a cache-oblivious sorting algorithm
Brodal, Fagerberg, Vinther (2008)

Sualeh Asif

MIT Computer Science  Artificial Intelligence Laboratory
Massachusetts Institute of Technology
CACHE-OBVIOUS SORTING
CACHE-OBVIOUS SORTING: A Timeline

Funnel Sorting
(Frigo et.al 1999)
CACHE-OBVIOUS SORTING: A Timeline

Funnel Sorting
(Frigo et al.; 1999)

Lazy Funnel Sorting
(Brodal, Fageberg; 2002)
CACHE-OBVIOUS SORTING: A Timeline

Funnel Sorting
(Frigo et al.; 1999)

Lazy Funnel Sorting
(Brodal, Fageberg; 2002)

Engineered Cache-Oblivious Sort
(Brodal, Fagerberg, Vinther; 2008)
**Cache-Oblivious Sorting: A Timeline**

Funnel Sorting
(Frigo et al.; 1999)

Lazy Funnel Sorting
(Brodal, Fagerberg; 2002)

Engineered Cache-Oblivious Sort
(Brodal, Fagerberg, Vinther; 2008)

**Theorem 1 (Cache-Oblivious Sorting)**

N items, M (small) memory blocks.
Funnel sorting can sort the N items in time:

\[ \Theta \left( \frac{N}{B} \log_{\frac{M_B}{N/B}} \frac{N}{B} \right) \]

This is optimal under reasonable assumptions.
Cache-Oblivious Sorting

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory B blocks.
Funnel sorting time:
\( \Theta \left( \frac{N}{B} \log \frac{N}{B} \right) \)
Thin Cache Oblivious Sorting

Thm 1 (Cache-Oblivious Sorting)
\[ \text{N items, } M \text{ (small) memory, } B \text{ blocks.} \]
Funnel sorting time:
\[ \theta \left( \frac{N}{B} \log_{\frac{N}{B}} \right) \]

Why Care ...?
**CACHE-OBVIOUS SORTING**

Why Care ...?

* Sort is (almost surely!!) fundamental problem  
  $\Rightarrow$ fast sorting implementation independent  
  of (constantly) changing memory hierarchy!

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory  
B blocks.  
Funnel sorting time:

$\Theta\left(\frac{N}{B} \log \frac{N}{B}\right)$
**Cache-Oblivious Sorting**

Why Care ...?

* Sort is (almost surely!!) fundamental problem
  => fast sorting implementation independant of (constantly) changing memory hierarchy!

* Experimentally check which heuristics are important

Thm 1 (Cache-Oblivious Sorting)

$N$ items, $M$ (small) memory

$B$ blocks.

Funnel sorting time:

$\Theta\left(\frac{B}{B} \log \frac{N}{B}\right)$
CACHE-OBVIOUS SORTING

Why Care ...?

* Sort is (almost surely!!) fundamental problem
  => fast sorting implementation independent of (constantly) changing memory hierarchy!

* Experimentally check which heuristics are important

* Curious humans ... and it is cool!

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:
\( \Theta \left( \frac{N}{B} \log \frac{N}{B} \right) \)
CACHE-OBLIVIOUS SORTING

Why Care ...?

* Sort is (almost surely!!) fundamental problem
  ⇒ fast sorting implementation independent of (constantly) changing memory hierarchy!
* Experimentally check which heuristics are important
* Curious humans ... and it is cool!

Punchline: near the best regardless !!

(Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:
\[ \Theta \left( \frac{N}{B} \log \frac{N}{MB} \right) \]
Cache-Oblivious Sorting

Why Care ...?

* Sort is (almost surely!!) fundamental problem ⇒ fast sorting implementation independent of (constantly) changing memory hierarchy!

* Experimentally check which heuristics are important

* Curious humans ... and it is cool!

Questions?
**Funnel Sorting: A Recap**

- **Primitives:**
  - Van Emde Boas Layout
  - Binary Mergers (and trees)
  - Lazy merging

- **Thm 1 (Cache-Oblivious Sorting):**
  - N items, M (small) memory
  - B blocks
  - Funnel sorting time: $\Theta(\frac{N}{B} \log_{M/B} N/B)$

- Not actually by VEB.
Funnel-Sorting: A Recap

Van Emde Boas Layout

BBST

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory blocks.
Funnel sorting time:
\[ \Theta \left( \frac{N}{B} \log \frac{N}{MB} \right) \]

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
Funnel-Sorting: A Recap

Van Emde Boas Layout

BBST - N elements

\( \{ \frac{1}{2} \log N \} \)

\( \log N \)

\( \Theta \left( \frac{N}{B} \log \frac{N/B}{B} \right) \)

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory
B blocks.
Funnel sorting time:
Funnel - Sorting: A Recap

Van Emde Boas Layout

BBST - N elements

Recursive partitioning

Recursively layout: \( \sqrt{N} \)

\( \{ \frac{1}{2} \lg N \} \cup \lg N \)

\[ \Theta \left( \frac{N}{B} \log \frac{N}{B} \right) \]

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory
B blocks
Funnel sorting time:

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
Funnel - Sorting: A Recap

Van Emde Boas Layout

BBST - N elements

Recursively layout:

Thm 1 (Cache-Oblivious)

N items, M (small) memory
B blocks.
Funnel sorting time:

\[ \Theta\left(\frac{B}{N} \log_{M/B} N/B\right) \]

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
Funnel - Sorting: A Recap

Van Emde Boas Layout

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:
\[ \Theta(\frac{N}{B} \log_{\frac{M}{B}} N / B) \]

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
Funnel-Sorting: A Recap

Van Emde Boas Layout

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:
\[ \Theta\left(\frac{N}{B} \log \frac{N}{MB}\right) \]

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
FUNNEL - SORTING : A RECAP

Van Emde Boas Layout

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory B blocks.
Funnel sorting time:
$\Theta\left(\frac{N}{B} \log \frac{N}{BM}\right)$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
Funnel - Sorting: A Recap

Van Emde Boas Layout

Theorem 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:
$\Theta\left(\frac{N}{B} \log \frac{N}{M_B} \right)$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
Funnel-Sorting: A Recap

Van Emde Boas Layout

Thm 1 (Cache-Oblivious)

N items, M (small) memory
B blocks.
Funnel sorting time:
\[ \Theta \left( \frac{N}{B} \log \frac{N}{B} \right) \]

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
Funnel-Sorting: A Recap

Van Emde Boas Layout

- Fast in memory
- Consecutive in memory

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks
Funnel sorting time:
$\Theta(\frac{B}{m} \log \frac{N}{B})$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
FUNNEL - SORTING: A RECAP

Binary Merging

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

\[ \Theta \left( \frac{B}{b} \log \frac{N}{b} \right) \]

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
Funnel Sorting: A Recap

Binary Merging

Input Buffers → Merge Step → Output Buffers

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks
Funnel sorting time:
\[ \Theta(B \log \frac{N}{B}) \]

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
**Funnel Sorting: A Recap**

**Binary Merging**

**Merge Step**: Input Buffers → Output Buffers.

**Idea**: Chain together to a tree.

**Theorem 1 (Cache-Oblivious Sorting)**

- N items, M (small) memory
- B blocks
- Funnel sorting time: \( \Theta \left( \frac{N}{B} \log \frac{N}{M_B} \right) \)

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
Funnel Sorting: A Recap

Binary Merging

Merge Step

Input Buffers

Output Buffer

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:

$\Theta(\frac{N}{B} \log \frac{N}{B})$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
Funnel - Sorting: A Recap

Binary Merging

Input Buffers

Output Buffer

Merge Step

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory
B blocks.
Funnel sorting time:

$\Theta(\frac{N}{B} \log \frac{N}{B})$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
FUNNEL - SORTING: A RECAP

Binary Merging

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks
Funnel sorting time:
\[ \Theta \left( \frac{N}{B} \log \frac{N}{B} \right) \]

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
**Funnel Sorting: A Recap**

**Binary Merging**

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory
B blocks.
Funnel sorting time:

\[ \Theta \left( \frac{N}{B} \log \frac{N}{B} \right) \]

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
**Funnel-Sorting: A Recap**

**Binary Merging**

- **Merge Step**
- **Input Buffers**
- **Output Buffers**

**Theorem 1 (Cache-Oblivious Sorting)**

N items, M (small) memory

B blocks.

Funnel sorting time:

$$\Theta \left( \frac{N}{B} \log \frac{M}{B} \right)$$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging

**Diagram:**

- **Input Buffers**
- **Output Buffers**
- **Mergers**
- **Size = \( k^3 \)**
- **Size = \( \alpha \left[ d^{3/2} \right] \)**

- **Input Buffers:**
  - \( \log k \)
  - \( \log k - 1 \)

- **Output Buffers:**
  - \( \log k \)

- **Mergers:**
  - \( k-1 \) mergers
Funnel-Sorting: A Recap

Lazy Merging

Merges will be lazy:

Theorem 1 (Cache-Oblivious Sorting)

N items, M (small) memory, B blocks.

Funnel sorting time:

\[ \Theta \left( \frac{N}{B} \log \frac{N}{B} \right) \]

- Van Emde Boas Layout
- Binary Mergers (and trees)
- Lazy merging
Funnel Sorting: A Recap

Lazy Merging

Merges will be lazy:

Procedure $\text{FILL}(v)$

while $v$’s output buffer is not full
  if left input buffer empty
    $\text{FILL}$(left child of $v$)
  if right input buffer empty
    $\text{FILL}$(right child of $v$)
  perform one merge step

Fig. 2. The merging algorithm.
**Funnel Sorting: A Recap**

Lazy Merging

Merges will be lazy:

Lazy: fill up an buffer only if it runs empty. (avoid silly checks)

Procedure **FILL**($v$)

while $v$’s output buffer is not full
  if left input buffer empty
    FILL(left child of $v$)
  if right input buffer empty
    FILL(right child of $v$)
perform one merge step

Fig. 2. The merging algorithm.

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory
B blocks
Funnel sorting time:

$\Theta \left( \frac{B}{N^2} \log \frac{N}{B} \right)$

* Van Emde Boas Layout
* Binary Mergers (and trees)
* Lazy merging
**Funnel Sorting: An Engineer's View**

**Theorem 1 (Cache-Oblivious Sorting)**

- $N$ items, $M$ (small) memory, $B$ blocks.
- Funnel sorting time: $\Theta\left(\frac{N}{B} \log\frac{N}{B}\right)$

**Funnel Sorting**

- **Primitives:**
  - Van Emde Boas Layout
  - Binary Mergers (and trees)
  - Lazy merging

![Funnel Sorting Diagram](Image)
**Funnel-Sorting: An Engineer's View**

- **Layout:**
  - DFS
  - BFS
  - VEB

- **Primitives:**
  - Van Emde Boas Layout
  - Binary Mergers (and trees)
  - Lazy merging

**Theorem 1 (Cache-Oblivious Sorting):**

- **N items, M (small) memory**
- **B blocks**

Funnel sorting time: \( \Theta \left( \frac{N}{B} \log \frac{N}{MB} \right) \)
**Funnel-Sorting: An Engineer's View**

- **Layout:**
  - DFS
  - BFS
  - VEB

- **Merging:**
  - Naive
  - Hybrid

- **Funnel Sorting**

- **Primitives:**
  - Van Emde Boas Layout
  - Binary Mergers (and trees)
  - Lazy merging

- **Thm 1 (Cache-Oblivious Sorting):**
  - N items, M (small) memory
  - B blocks.
  - Funnel sorting time:
    \[ \Theta \left( \frac{N}{B} \log \frac{N}{B} \right) \]

- **Funnel Sorting Diagram:**
  - Input buffers
  - Output buffer
**Funnel Sorting**: An Engineer's View

**Theorem 1 (Cache-Oblivious Sorting)**

- N items, M (small) memory
- B blocks
- Funnel sorting time:
  \[ \Theta \left( \frac{B}{8} \log \frac{N}{B} \right) \]

- **Layout**:
  - DFS
  - BFS
  - VEB

- **Merging**:
  - Naive
  - Hwang, Lin, Knuth
  - Hybrid
  - B-way merges
    - Value of B

- **Primitives**:
  - Van Emde Boas Layout
  - Binary Mergers (and trees)
  - Lazy merging

- **Funnel Sorting**
Funnel-Sorting: An Engineer's View

- Layout:
  - DFS
  - BFS
  - VEB

- Merging:
  - Naive
  - Hwang, Lin, Knuth
  - Hybrid

- Funnel Sorting
  - B-way merges
    - Value of B

- Primitives:
  - Van Emde Boas Layout
  - Binary Mergers (and trees)
  - Lazy merging

- Base Sort:
  - Insertion
  - Selection
  - HeapSort
  - Shell Sort
  - Quick Sort
  - std::sort

- Theorem 1 (Cache-Oblivious Sorting):
  - N items, M (small) memory
  - B blocks
  - Funnel sorting time: \( \Theta\left(\frac{B}{b} \log \frac{N}{B}\right) \)

- Hyper parameters:
  - \( \alpha, \beta \)
  - in buffer size \( \alpha k^d \)
Funnel-Sorting: An Engineer's View

Idea: Try Everything!!

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:
$\Theta(\frac{N}{B} \log_{\frac{N}{MB}} \frac{N}{B})$

funnel sorting:
Funnel - Sorting: An Engineer's View

Method:

* Try different processors: { Pentium 3+4, MIPS, AMD, Titanium }

* data distributions

* data types:
  - ints
  - ints + ptrs
  - records (100 bytes)

* randomness:
  - drand48 -> C lib

Theorem 1 (Cache-Oblivious Sorting):

\[ T_n = \Theta(n \log \frac{N}{B}) \]

Funnel sorting time:
**Funnel-Sorting: An Engineer's View**

- **Layout:**
  - DFS
  - BFS
  - VEB

- **Merging:**
  - Naive
  - Hwang, Lin, Knoth
  - Hybrid

- **Base Sort:**
  - Insertion
  - Selection
  - Heap Sort
  - Shell Sort
  - Quick Sort
  - std::sort

- **Primitives:**
  - Van Emde Boas Layout
  - Binary Mergers (and trees)
  - Lazy merging

- **Funnel Sorting:**

- **Hyperparameters:**
  - $\alpha$, $d$
  - in buffer size $\alpha K^d$

**Theorem 1 (Cache-Oblivious Sorting):**

- $N$ items, $M$ (small) memory
- $B$ blocks
- Funnel sorting time:
  $$\Theta\left(\frac{B}{\alpha} \log \frac{N}{B}\right)$$
**Funnel - Sorting: An Engineer's View**

**Layout:**
- DFS
- BFS
- VEB

**Primitives:**
- Van Emde Boas Layout
- Binary Merges (and trees)
- Lazy merging

**Base Sort:**
- Insertion
- Selection
- Heap Sort
- Shell Sort
- Quick Sort
- std::sort

**Cache-Oblivious Sorting**

N items, M (small) memory

B blocks

Funnel sorting time:

\[ \Theta\left(\frac{N}{B} \log \frac{N}{B}\right) \]

**Hyper parameters**

- \(x, d\)

in buffer sz.

- \(ak^d\)

**Merging:**
- Naive
- Hwang, Lin, Knuth
- Hybrid

- B-way merges
- Value of B

**Funnel Sorting**

- Winner: \( \approx 65\% \)
- VEB + std. allocator + nodes and buffers separate
**Funnel - Sorting** : An Engineer's View

- **Layout**:
  - DFS
  - BFS
  - vEB

- **Primitives**:
  - Van Emde Boas
  - Binary Mergers (and trees)
  - Lazy merging

- **Base Sort**:
  - Insertion
  - Selection
  - Heap Sort
  - Shell sort
  - Quick Sort
  - std :: sort

- **Funnel Sorting**

- **Hyper parameters**:
  - $\alpha$, $\delta$

- **Theorem 1 (Cache-Oblivious Sorting)**
  - $N$ items, $M$ (small) memory
  - $B$ blocks
  - Funnel sorting time:
  - $\Theta\left(\frac{2}{B} \log \frac{N}{B}\right)$

- **Merging**:
  - naive
  - Hwang, Lin, Knuth
  - Hybrid

- **Winner**:
  - naive
  - (bc branch) pred.
  - $B$-way merges
  - value of $B$
Funnel-Sorting: An Engineer's View

Merging:
- naive
- Hwang, Lin, Knuth
- Hybrid

Funnel Sorting

Layout:
- DFS
- BFS
- VEB

Primitives:
- Van Emde Boas Layout
- Binary Mergers (and trees)
- Lazy merging

Base Sort
- Insertion
- Selection
- Heap Sort
- Shell Sort
- Quick Sort
- std::sort

Thm 1 (Cache-Oblivious Sorting)

N items, M (small) memory
B blocks.
Funnel sorting time:

\[ \Theta \left( \frac{B}{8} \log \frac{N}{8} \right) \]
Funnel Sorting: An Engineer's View

Merging:
- naive
- Huang, Lin, Knuth
- Hybrid

Funnel Sorting
- Layout:
  - DFS
  - BFS
  - vEB
- B-way merges
- value of B

Primitives:
- Van Emde Boas
  - Layout
- Binary Mergers (and trees)
- Lazy merging

Base Sort
- Insertion
- Selection
- HeapSort

Winner: std::sort (before 400)

Hyperparameters
- $\alpha, d$
  - in buffer size
  - $d K^2$

Thm 1 (Cache-Oblivious Sorting)
- $N$ items, $M$ (small) memory
- $B$ blocks
- Funnel sorting time:
  - $\Theta(\frac{B}{8} \log_2 \frac{N}{B})$
**Funnel-Sorting: An Engineer's View**

N items, M (small) memory, B blocks.

**Thm 1 (Cache-Oblivious Sorting)**

Funnel sorting time:

\[ \Theta \left( \frac{N}{B} \log_{N_B} \frac{N}{B} \right) \]

**Funnel Sorting**

- **Layout:**
  - DFS
  - BFS
  - VEB

- **Primitives:**
  - Van Emde Boas Layout
  - Binary Mergers (and trees)
  - Lazy merging

- **Merging:**
  - Naive
  - Hybrid
  - Hwang, Lin, Knuth

- **Base Sort**
  - Insertion
  - Selection
  - HeapSort

- **Winner:** \((\alpha \frac{N}{4})\times\alpha, d\) in buffer size \(\alpha k^d\)

- **Hyper parameters**
  - \(\alpha = 16\)
  - \(k = 2\)
Funnel - Sorting: An Engineer's View

Thm 1 (Cache-Oblivious Sorting)

\[ T = \Theta(\frac{N}{B} \log \frac{N}{B}) \]

Input buffers

Output buffer

Funnel sorting:

- Binary Mergers (and trees)
- Lazy merging

Primitives:
- Van Emde Boas Layout
- Base Sort
- Insertion
- Selection
- HeapSort
- ShellSort
- std::sort

Hyperparameters:
- \( \alpha, \beta \)
- \( (16, 2) \) in buffer sz.
- \( \alpha K^2 \)

Merging:
- Naive
- Huang, Lin, Knuth
- Hybrid

Layout:
- DFS
- BFS
- VEB

B-way merges
- Value of B
  - 4

Funnel Sorting

Base Sort
FUNNEL-SORTING: AN ENGINEER'S VIEW

Vs Competitors:
* GCC std::sort
* Intel Dinkumware std::sort
* Bentley's sort
* (Cache-aware) TPIE sort
* (Cache-aware) Xiao et.al sort
* (Cache-aware) Arge et.al sort

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:
\[ \Theta \left( \frac{N}{B} \log \frac{N}{MB} \right) \]
Funnel Sorting: An Engineer's View

VS Competitors:

RAM:

Thm 1 (Cache-Oblivious Sorting)
N items, M (small) memory
B blocks.
Funnel sorting time:
$\Theta(\frac{N}{B} \log \frac{N}{B})$
FUNNEL - SORTING : AN ENGINEER'S VIEW

Thm 1 (Cache-Oblivious)

N items, M (small) memory
B blocks.
Funnel sorting time:
$\Theta\left(\frac{B}{8} \log \frac{N}{8}\right)$
Funnel-Sorting: An Engineer’s View

Vs Competitors:

Disk:

**Thm 1 (Cache-Oblivious Sorting)**

- N items, M (small) memory
- B blocks
- Funnel sorting time:
  \[ \Theta\left(\frac{3}{8} \log \frac{M}{B}\right) \]
Funnel Sorting: An Engineer's View

VS Competitors:

Disk:

Thm 1 (Cache-Oblivious)
Sorting
N items, M (small) memory
B blocks.
Funnel sorting time:
\[ \Theta \left( \frac{N}{B} \log \frac{N}{B} \right) \]
**Cache-Oblivious Sorting**

**Discussion Questions**

* Should we do (hard-core ablation-style) experiments for every algorithm?

* Does ablation style even work?

* Can these funneling ideas generalize to other interesting problems?

* Other architectures than a CPU

**Thm 1 (Cache-Oblivious Sorting)**

N items, M (small) memory blocks.

Funnel sorting time:

\[ \Theta \left( \frac{N}{B} \log \frac{N}{M/B} \right) \]
6.886 talk: Engineering a cache-oblivious sorting algorithm
Brodal, Fagerberg, Vinther (2008)

Sualeh Asif

MIT Computer Science Artificial Intelligence Laboratory
Massachusetts Institute of Technology