Cache-Oblivious Dynamic Dictionaries with Update/Query Tradeoff

Gerth Stølting Brodal
Erik D. Demaine
Jeremy T. Fineman
John Iacono
Stefan Langerman
J. Ian Munro

Result presented at SODA 2010

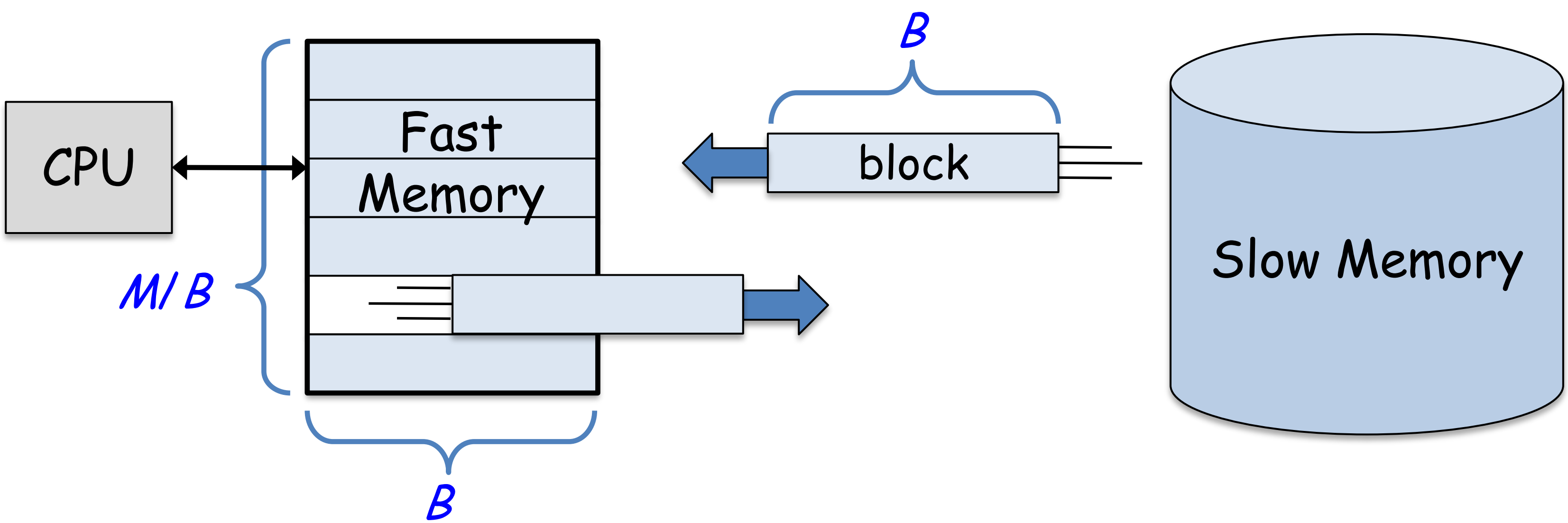
Dynamic Dictionary

Search(k)

Insert(e)
Delete(k)

I/O Model

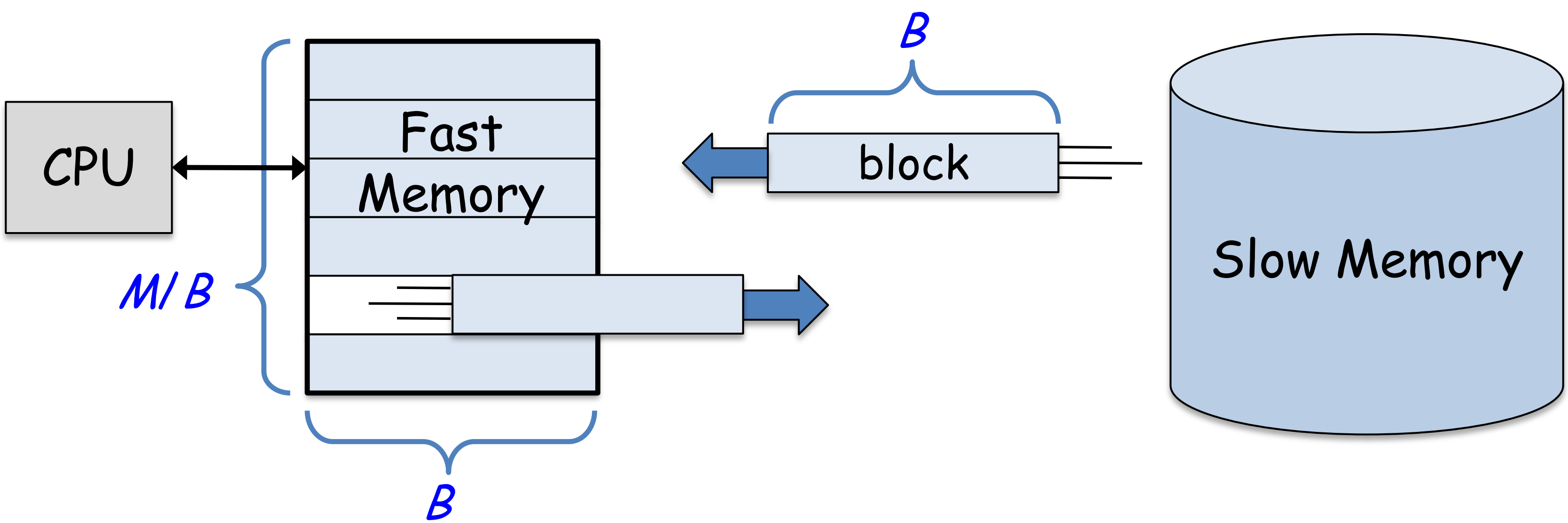
[Aggarwal, Vitter 88]



Cost: the number of ***block transfers*** (I/Os)

Cache-Oblivious Algorithms

[Frigo, Leiserson, Prokop, Ramachandran 99]



- Algorithms not parameterized by *M* or *B*
- Analyze in ***ideal-cache model*** — I/O model, except **optimal replacement** policy is assumed

Cache-Oblivious Dynamic Dictionaries

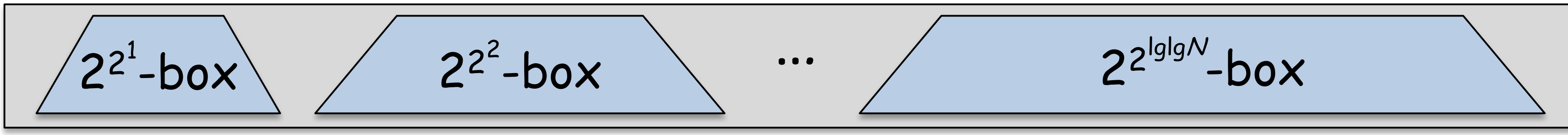
Cache-Aware	Search	Insert
B-tree [BM72]	$O(\log_B N)$	$O(\log_B N)$
Buffered B-tree [BF03]	$O((1/\varepsilon)\log_B N)$	$O((1/\varepsilon B^{1-\varepsilon})\log_B N)$*

Cache-Oblivious	Search	Insert
CO B-tree [BDF-00, BDIW04,BFJ02]	$O(\log_B N)$	$O(\log_B N + \ldots)$
COLA [BFF-CFKN07]	$O(\log_2 N)$	$O((1/B)\log_2 N)$*
Shuttle Tree [BFF-CFKN07]	$O(\log_B N)$	$O((1/B^{\Omega(1/(\log\log B)^2)})\log_B N + \ldots)$*
xDict [this paper]	$O((1/\varepsilon)\log_B N)$	$O((1/\varepsilon B^{1-\varepsilon})\log_B N)$*†

* amortized

† assumes $M = \Omega(B^2)$

Building an xDict (ε = 1/2)



$\lg\lg N$ **x-boxes** of squaring capacities

Insert: insert into smallest box

- When a box reaches capacity, **Flush** it and **Batch-Insert** into the next box
- $O((1/\sqrt{B})\log_B x)$ cost is dominated by largest box
  ➔ $O((1/\sqrt{B})\log_B N)$

Search: search in each x-box

- $O(\log_B x)$ cost is dominated by largest box $O(\log_B N)$

x-Box = **dictionary with capacity x^2**

Batch-Insert(D,A): insert $\Theta(x)$ presorted objects

— cost $O((1/\sqrt{B})\log_B x)$ per element

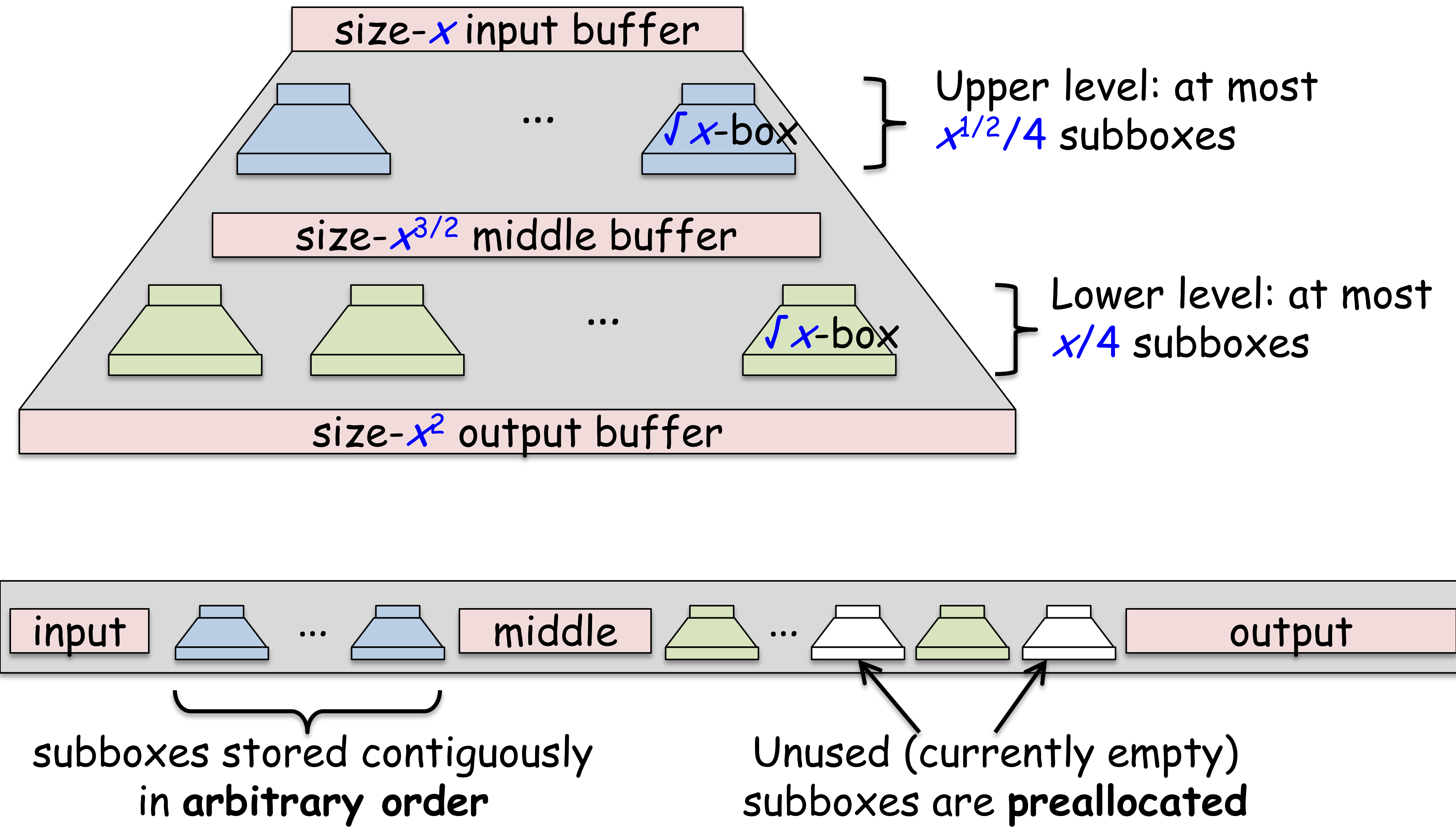
Search(D,K):

— cost is $O(\log_B x)$

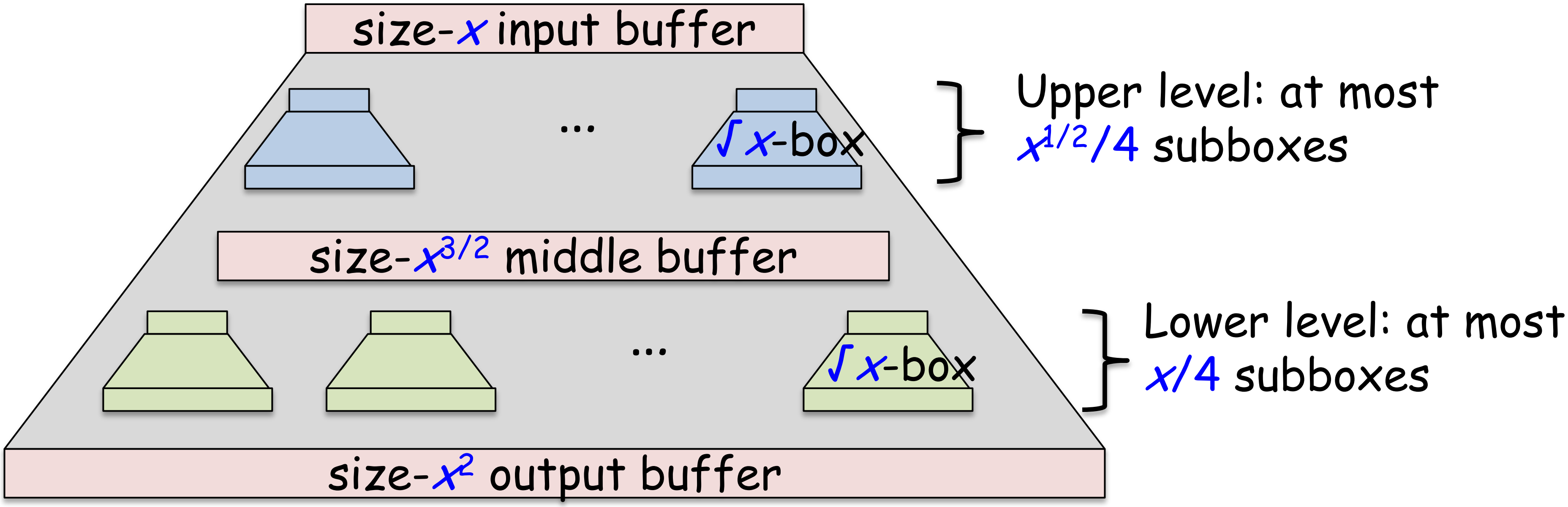
Flush(D): produce a size-x^2 sorted array A containing all the elements in the x-box D

— cost is $O(1/B)$ per element

Recursive x-Box



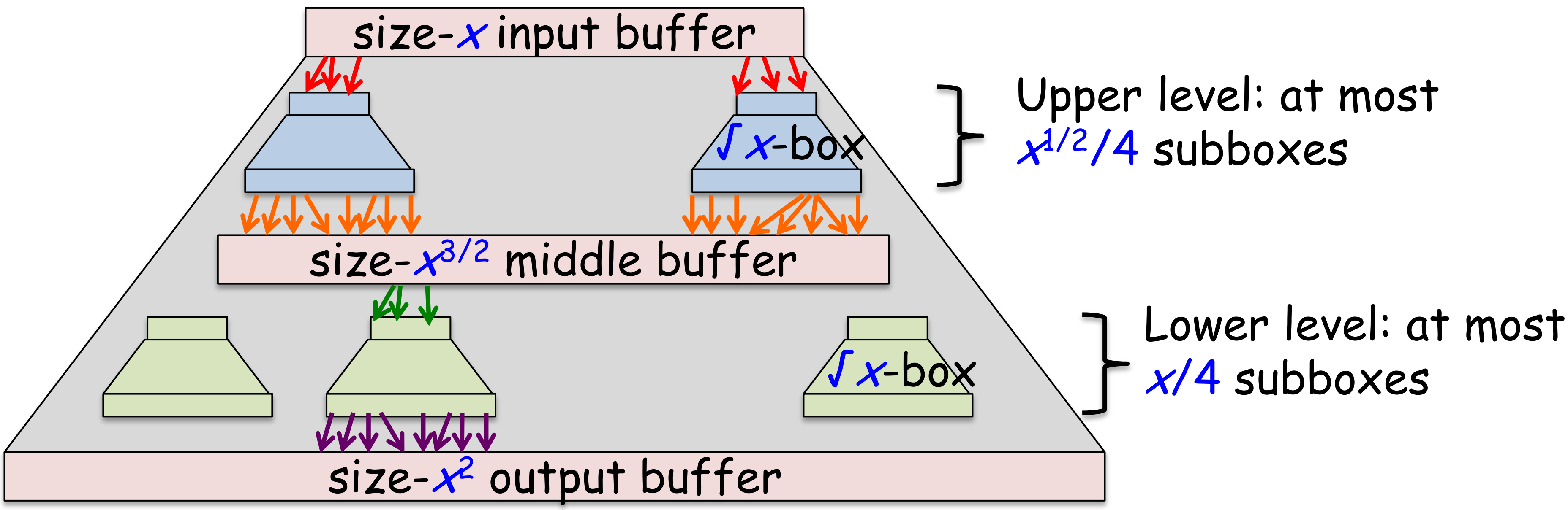
x-Box Space Usage



Theorem: An x-Box uses at most cx^2 space

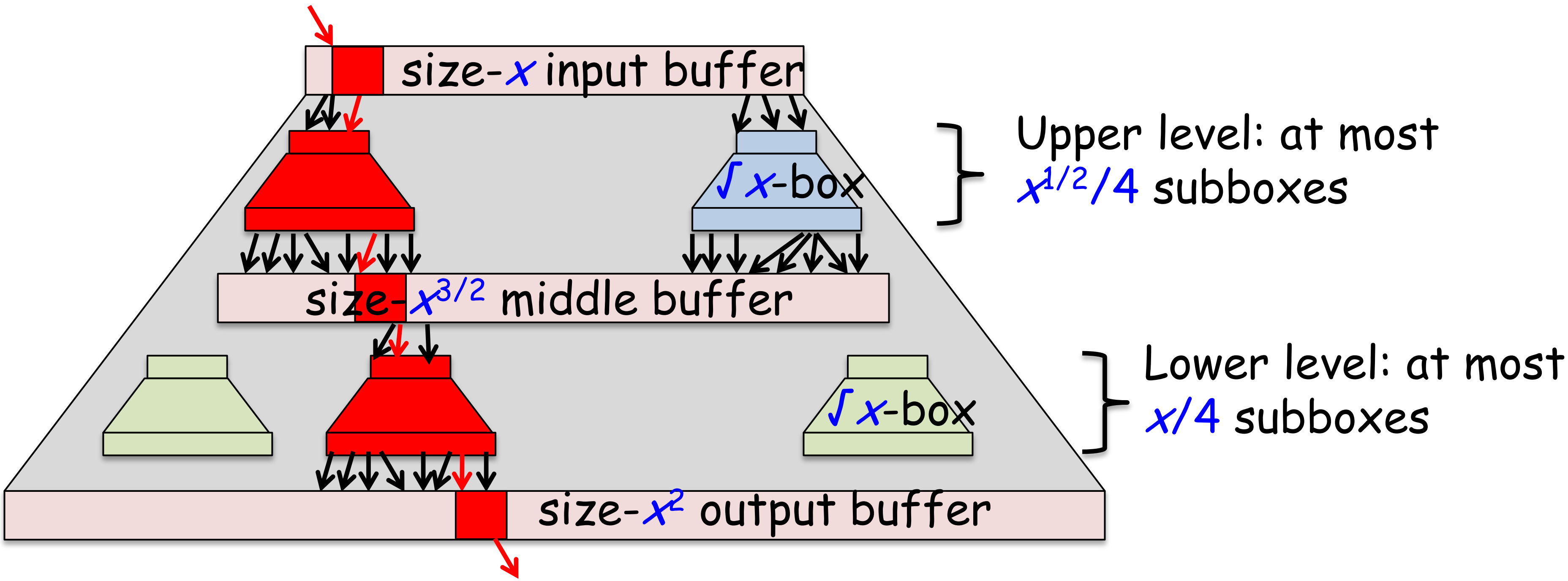
(within constant factor of capacity/output buffer)

Fractional Cascading within x-Box



Propagate samples upwards + Lookahead pointers

Searching in an x-Box



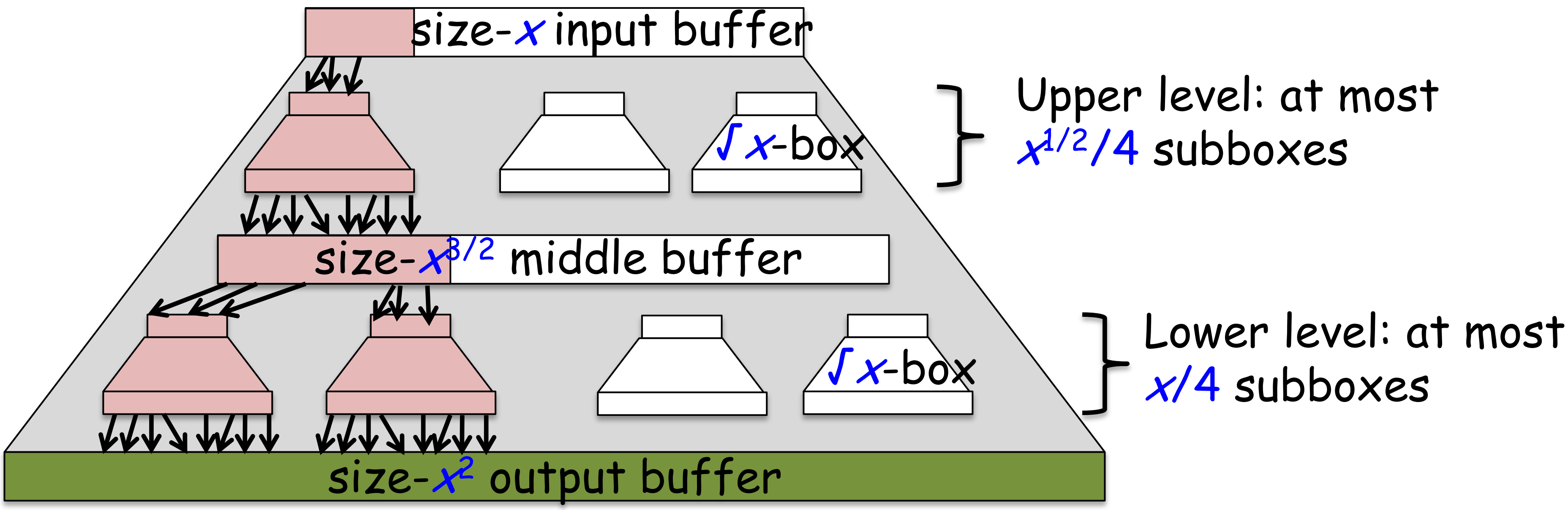
Describe searches by the recurrence

$$S(x) = 2S(\sqrt{x}) + O(1)$$

with base case $S(<\sqrt{B}) = 0$

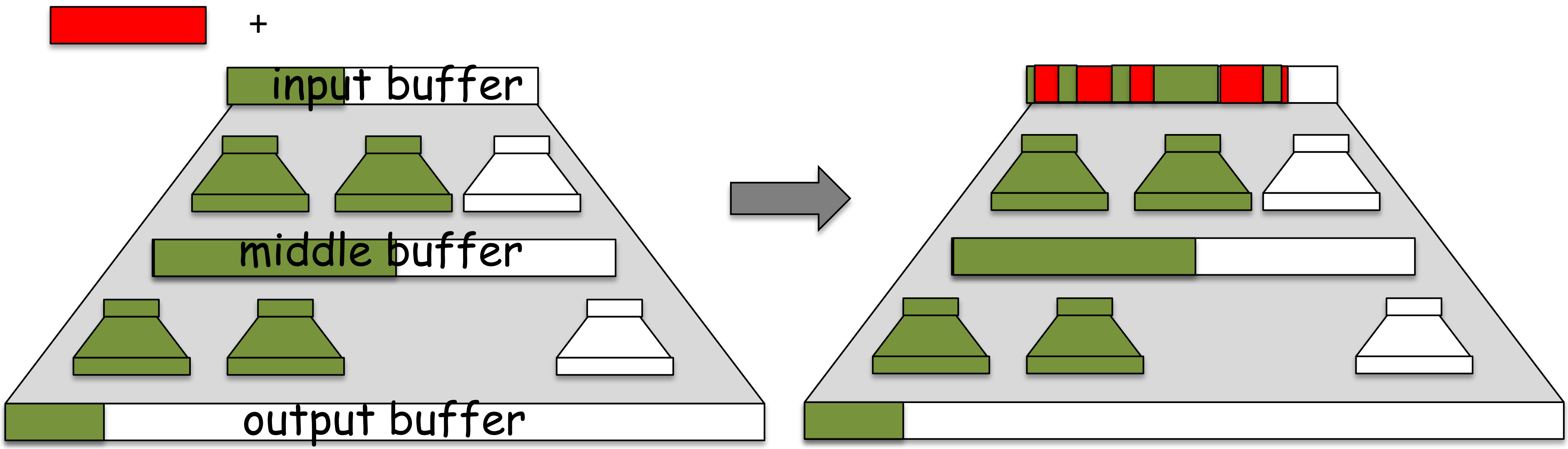
Solves to $O(\log_B N)$

Flush



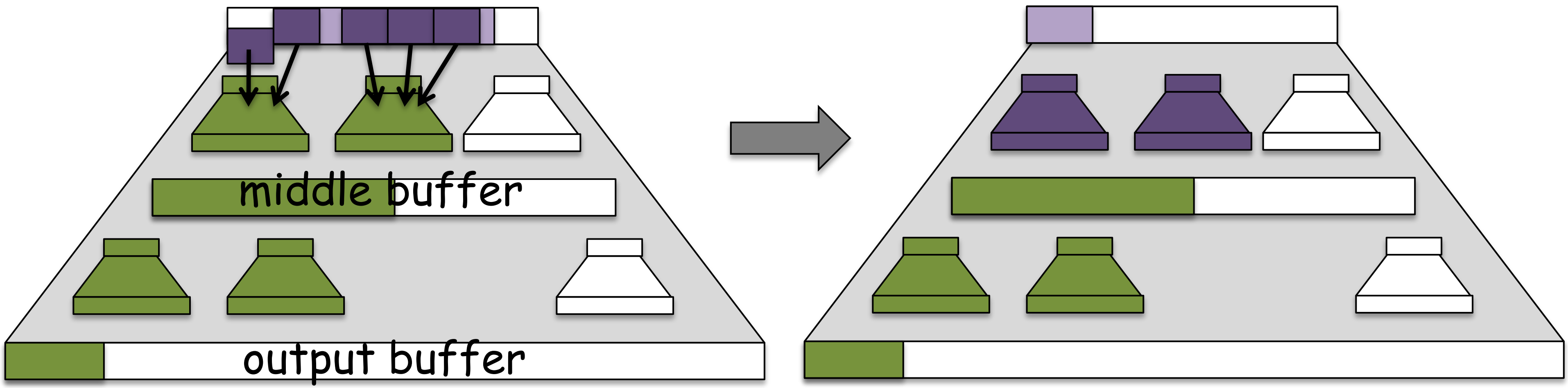
- Moves all real elements to the output buffer in sorted order.
- Lookahead pointers are rebuilt to facilitate searches. Most subboxes remain empty.

Batch-Insert



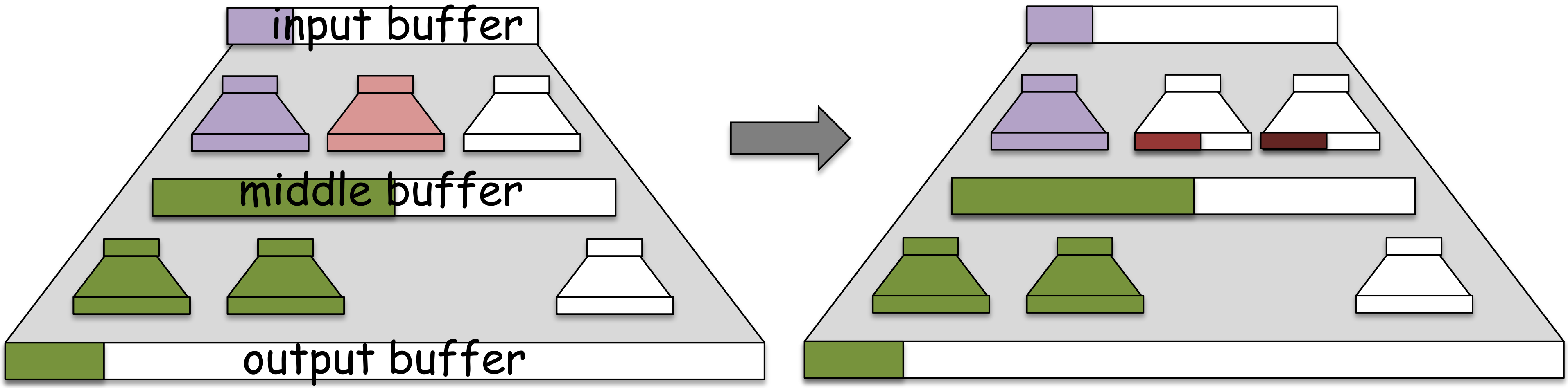
1. Merge sorted input into input buffer.

Batch-Insert

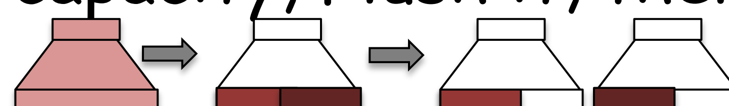


1. Merge sorted input into input buffer.
2. If input buffer is "full enough," Batch-Insert into upper-level subboxes (in chunks of $\Theta(\sqrt{x})$)

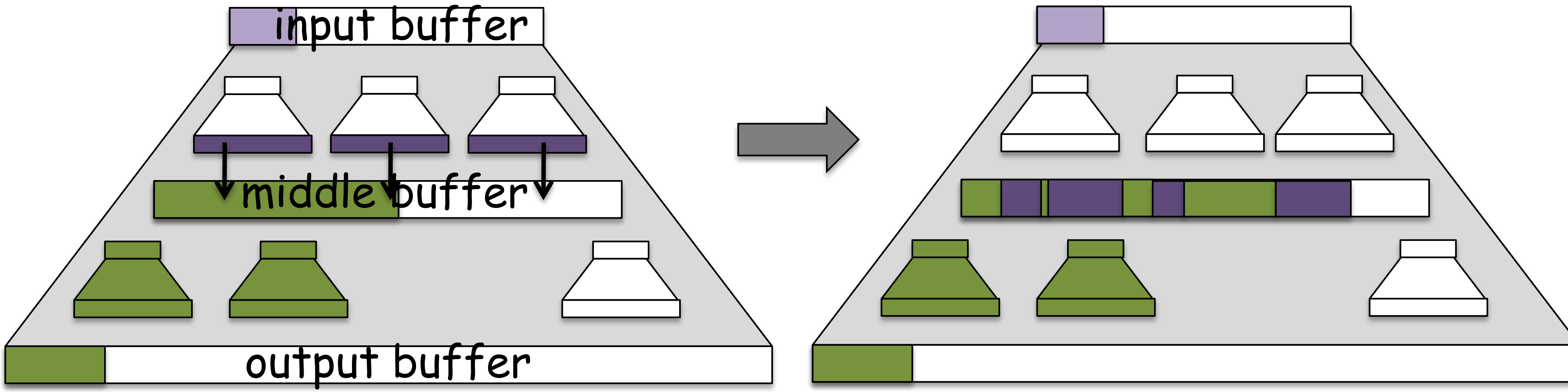
Batch-Insert



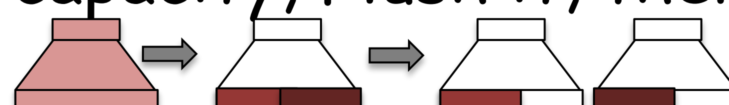
1. Merge sorted input into input buffer.
2. If input buffer is "full enough," Batch-Insert into upper-level subboxes (in chunks of $\Theta(\sqrt{x})$)
3. Whenever a subbox is near capacity, Flush it, then split it into two subboxes



Batch-Insert

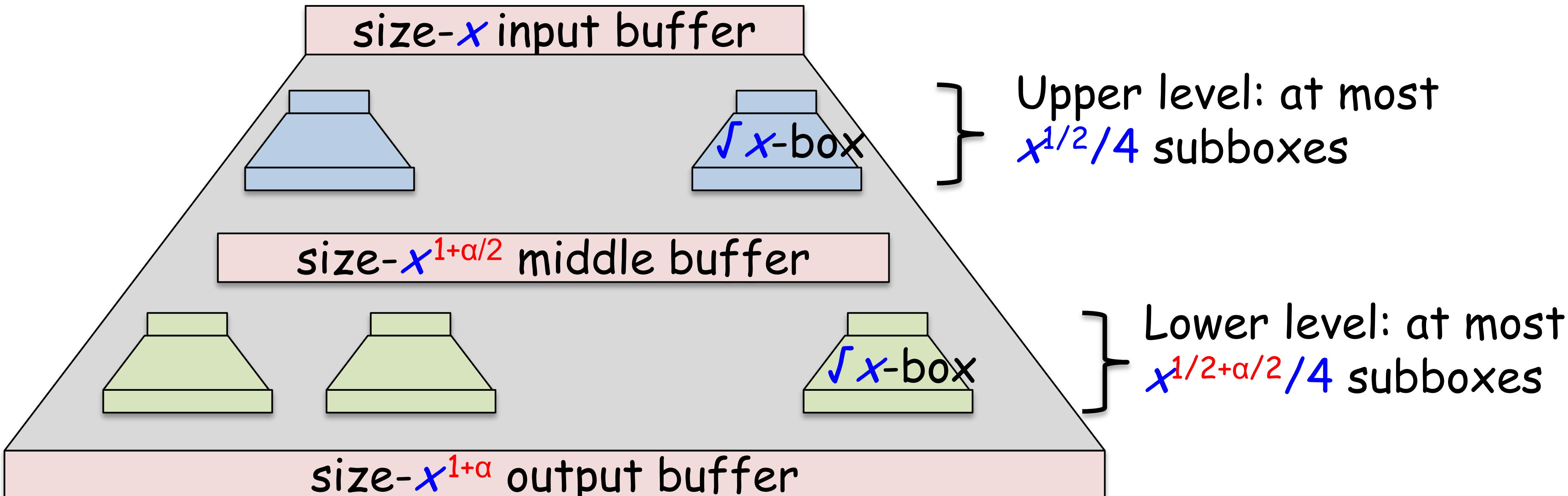


1. Merge sorted input into input buffer.
2. If input buffer is "full enough," Batch-Insert into upper-level subboxes (in chunks of $\Theta(\sqrt{x})$)
3. Whenever a subbox is near capacity, Flush it, then split it into two subboxes



4. If no empty subboxes remain, Flush all of them and merge output buffers into middle buffer.

Generalizing to $O((1/\varepsilon B^{1-\varepsilon})\log_B N)$



Parameterize by $0 < \alpha \leq 1$, where $\alpha = \varepsilon/(1-\varepsilon)$

$1/\varepsilon$ overhead comes from geometric sum in xDict

Results Summary

Cache-Aware	Search	Insert
B-tree [BM72]	$O(\log_B N)$	$O(\log_B N)$
Buffered B-tree [BF03]	$O((1/\varepsilon)\log_B N)$	$O((1/\varepsilon B^{1-\varepsilon})\log_B N)$*

Cache-Oblivious	Search	Insert
CO B-tree [BDF-00, BDIW04,BFJ02]	$O(\log_B N)$	$O(\log_B N + \ldots)$
COLA [BFF-CFKN07]	$O(\log_2 N)$	$O((1/B)\log_2 N)$*
Shuttle Tree [BFF-CFKN07]	$O(\log_B N)$	$O((1/B^{\Omega(1/(\log\log B)^2)})\log_B N + \ldots)$*
xDict [this paper]	$O((1/\varepsilon)\log_B N)$	$O((1/\varepsilon B^{1-\varepsilon})\log_B N)$*†

* amortized

† assumes $M = \Omega(B^2)$