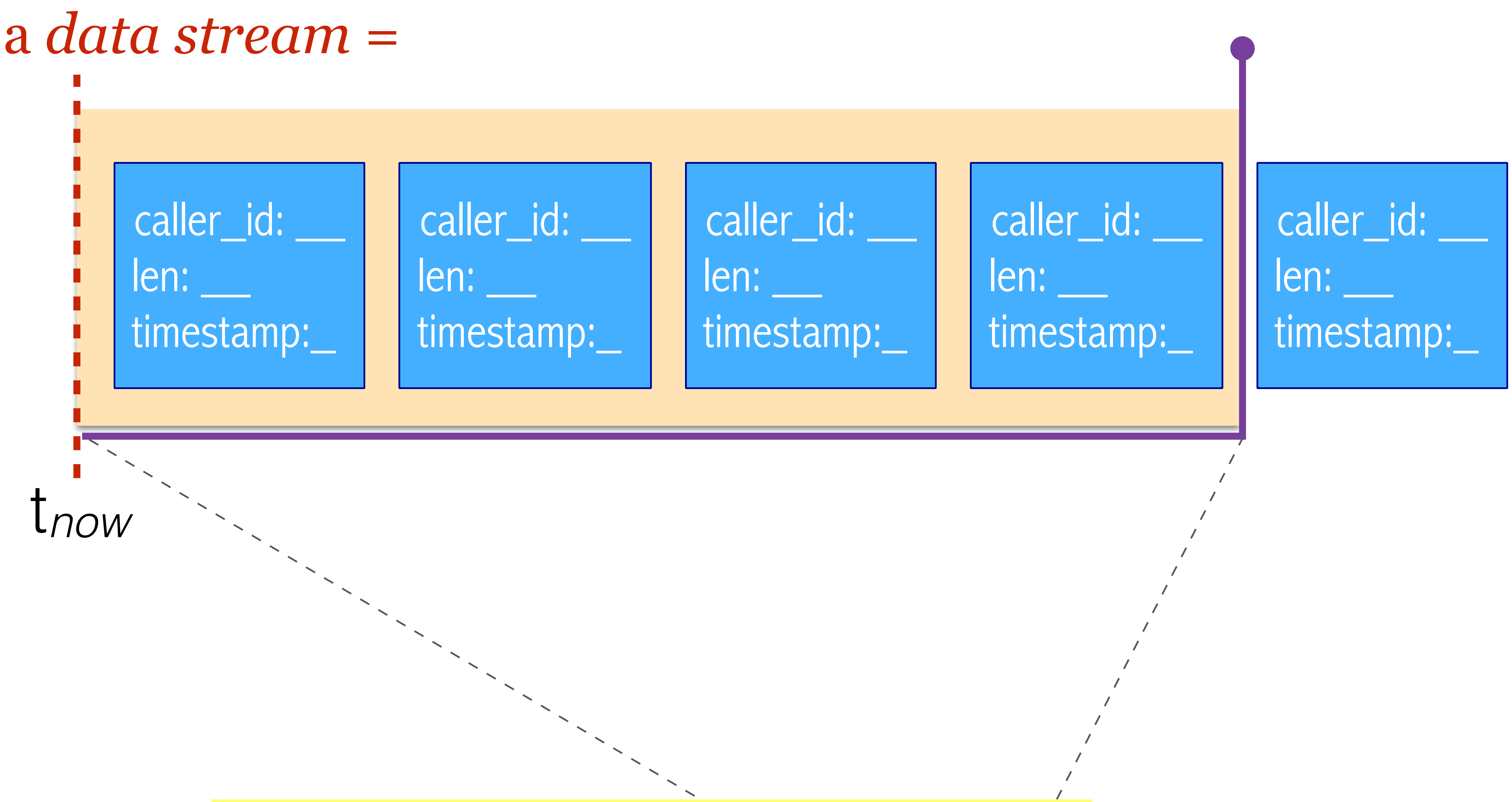
General Incremental Sliding-Window Aggregation

Kanat Tangwongsan

Mahidol University International College, Thailand
(part of this work done at IBM Research, Watson)

Joint work with Martin Hirzel, Scott Schneider, and Kun-Lung Wu
IBM Research, Watson

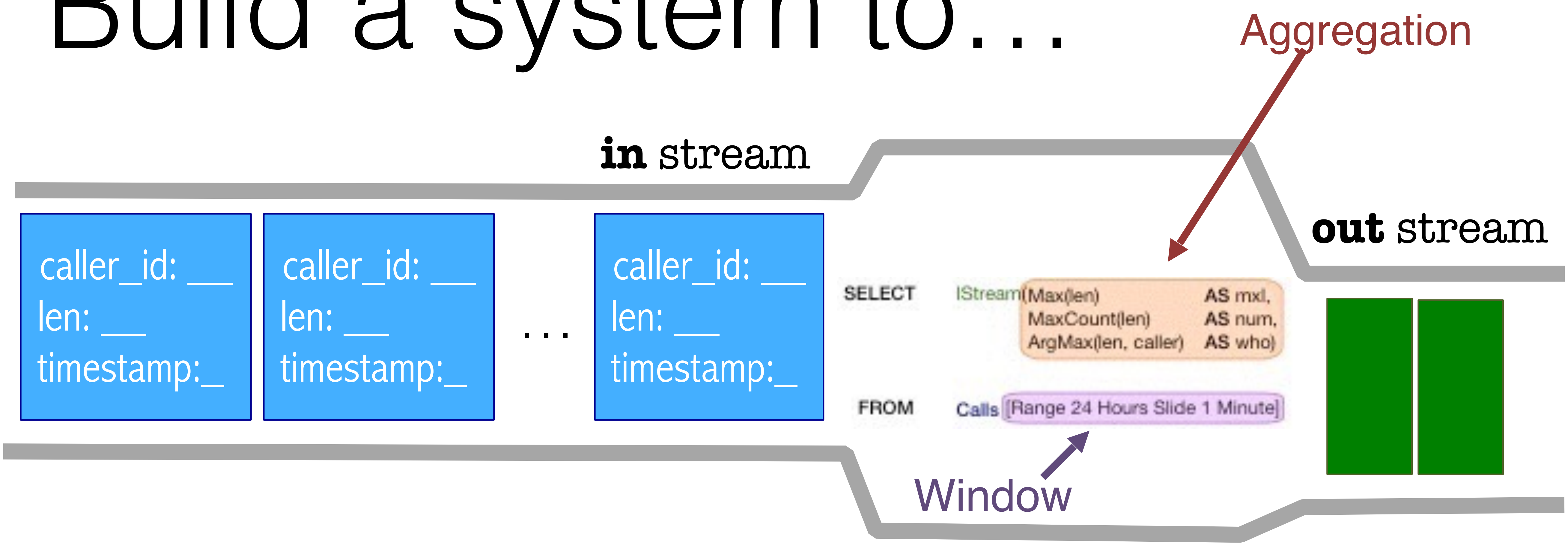
Dynamic data is everywhere



often interested in a **sliding window**

Compute some aggregation on this data

Build a system to…



Answer the following questions every minute about the past 24 hrs:

- How long was the longest call?
- How many calls have that duration?
- Who is a caller with that duration?

Answer the following questions every minute about the past 24 hrs:

- How long was the longest call?
- How many calls have that duration?
- Who is a caller with that duration?

Basic Solution:

Simple but slow:
$O(n)$ per query

Improvement Opportunities

Idea: When window slides, lots of common contents with the most recent query.

How to Reuse?

- If **invertible**, keep a running sum: add on arrival, subtract on departure

- **Partial sum**: bundle items that arrive and depart together to reduce # of items in the window.

This Work:

How to engineer sliding-window aggregation so that

- can add new aggregation operations easily
- can get good performance with little hassle

(using a combination of simple, known ideas)

Performance

Prior Work

- Good for small updates
  (e.g., Arasu-Widom VLDB'04, Moon et al. ICDE'00)
- Good for large updates
  (e.g., Cranor et al. SIGMOD'03, Krishnamurthi et al. SIGMOD'06)

This Work

Good for Large & Small:
If m updates are made to a window of size n, use $O(m\log(n/m))$ time to compute aggregate.

Generality

Prior Work

- Require invertibility or commutativity or assoc.
- Require FIFO windows

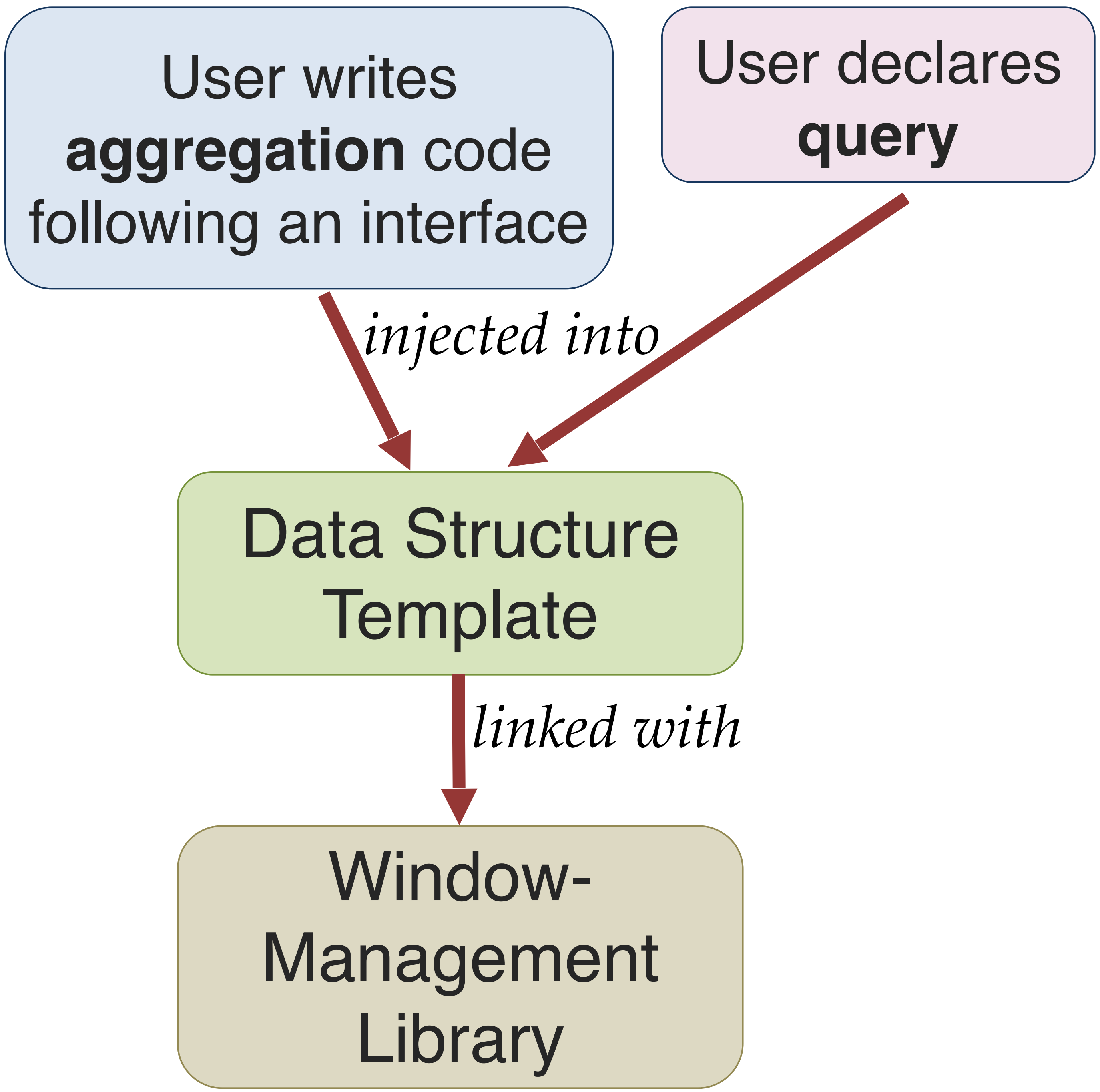
This Work

Require associativity (not but invertibility nor commutativity)

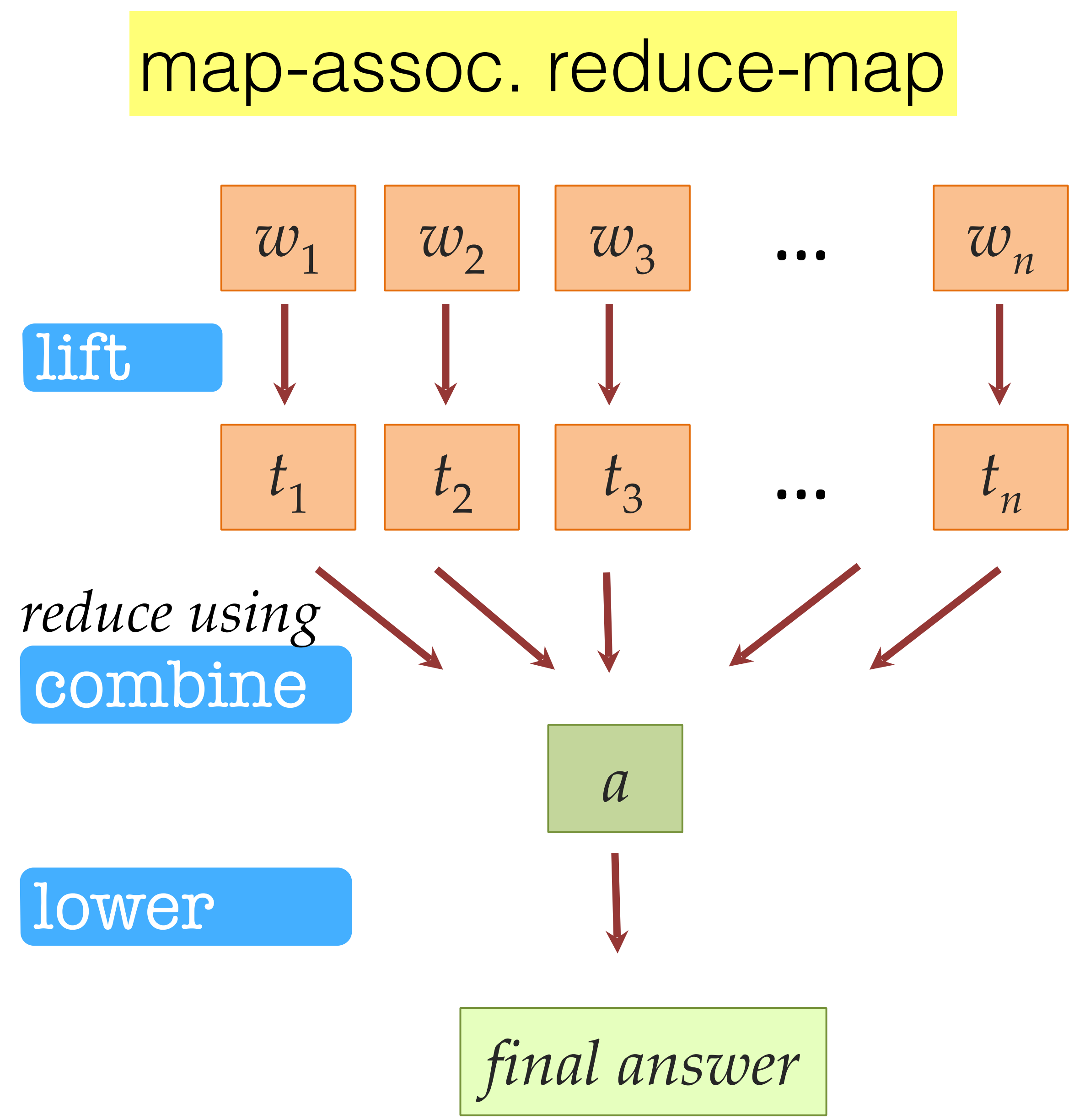
OK to be non-FIFO

Our Approach

High-level Idea



Aggregation Interface



Example

StdDev

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(w_i - \bar{x})^2}$$

count

sum of values

sum of squares

lift $x \mapsto \{c : 1, \Sigma : x, \sigma : x^2\}$

combine *component-wise add*

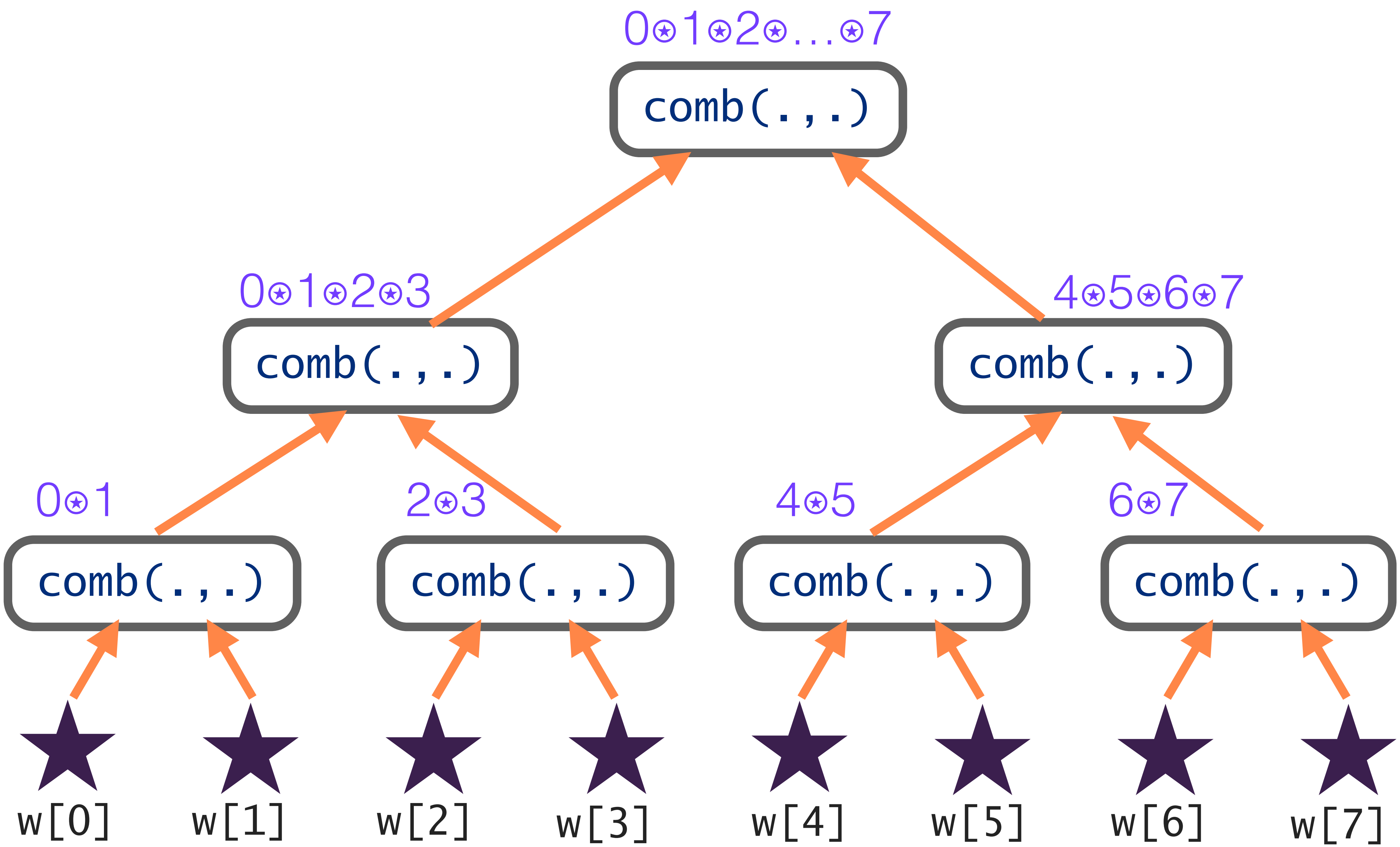
lower $\sqrt{\frac{1}{c}\left(\sigma - \Sigma^2/c\right)}$

Perfect Binary Tree

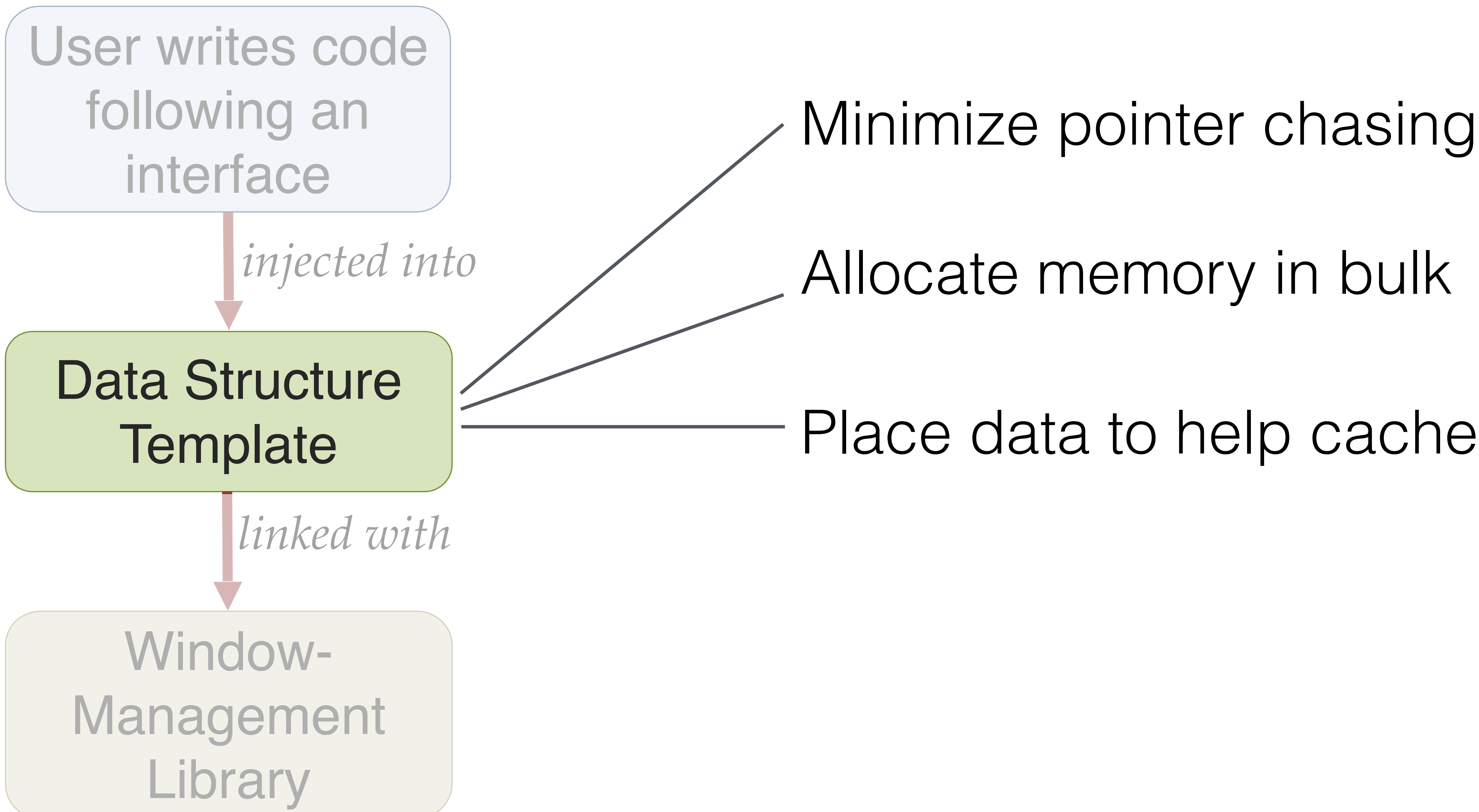
Keep Partial Sum in Internal Nodes

Depth: log n

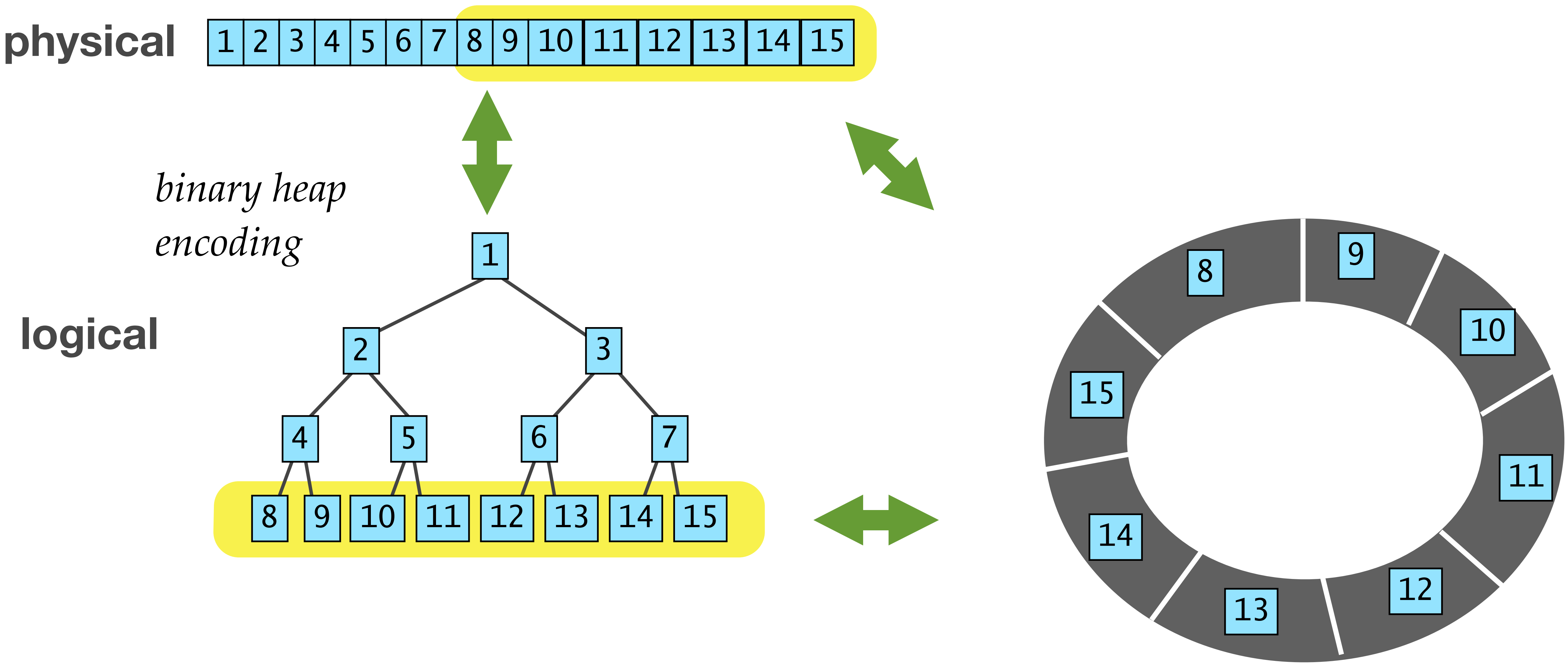
Changing k leaves affects $\leq O(k \log(n/k))$ nodes



Data Structure Engineering



Main idea:
keep a perfect binary tree, treating leaves as a circular buffer, all laid out as one array



Q: Non power of 2? Dynamic window size? non-FIFO?

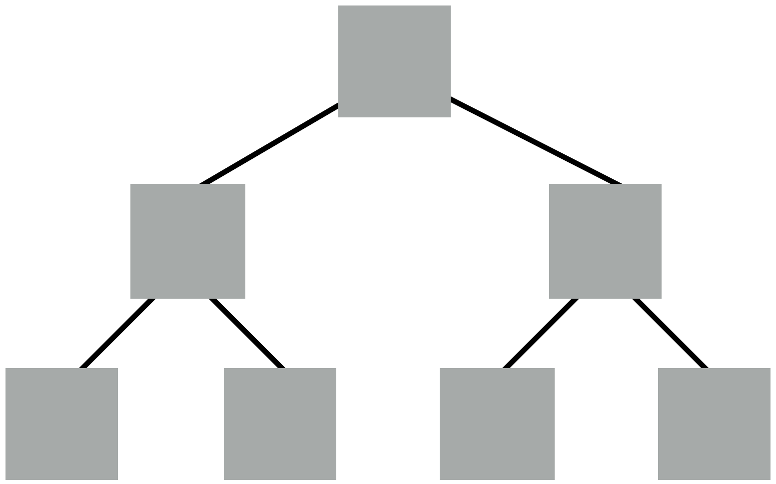
The queue’s front and back locations give a natural demarcation.

Resize and rebuild as necessary. Amortized bounds (see paper)

But…

Circular Buffer Leaves != Window-Ordered Data

leaf view



Fix? When inverted:

i j B F e f g h

prefix suffix

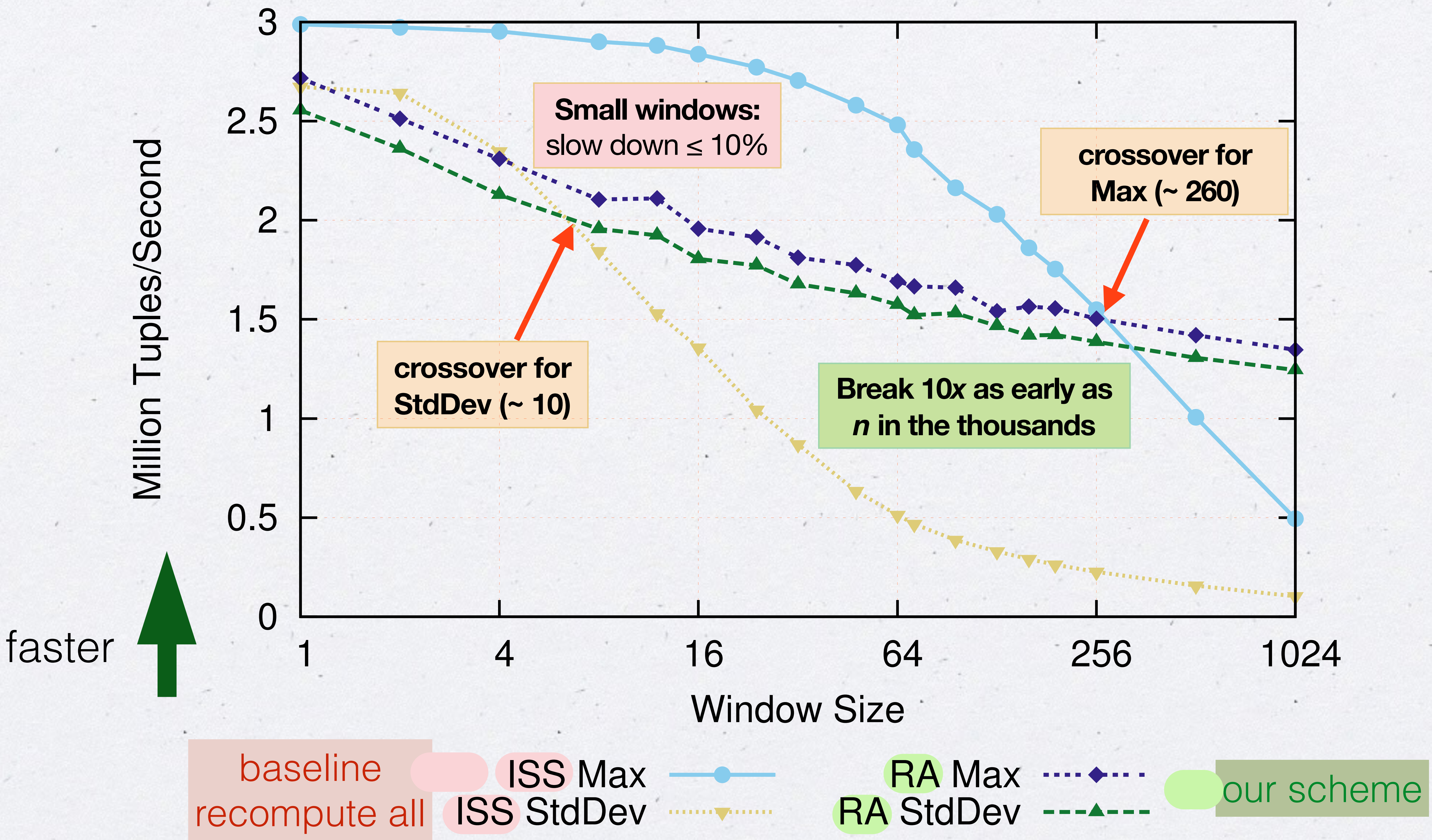
```
ans = combine(suffix, prefix)
```

F 1 B	1
F 1 2 B	1 2
F 1 2 3 B	1 2 3
F 1 2 3 4 B	1 2 3 4
F 2 3 4 B	2 3 4
5 B F 2 3 4	2 3 4 5

Experimental Analysis

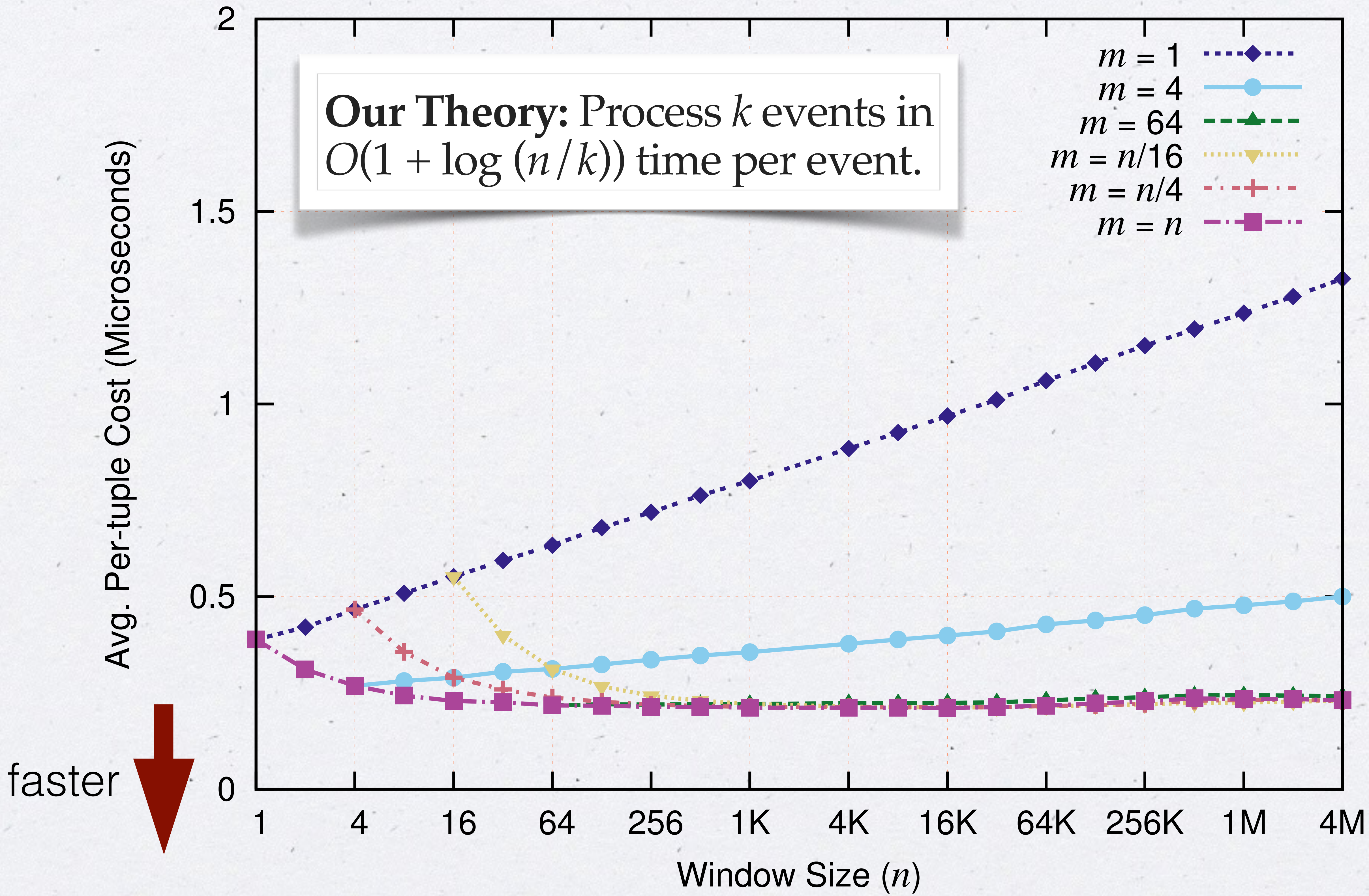
1. What’s the throughput relative to non-incremental?
2. What’s the performance trend under updates of different sizes?
3. How does wildly-changing window size affect the overall throughput?

1 What's the throughput relative to non-incremental?



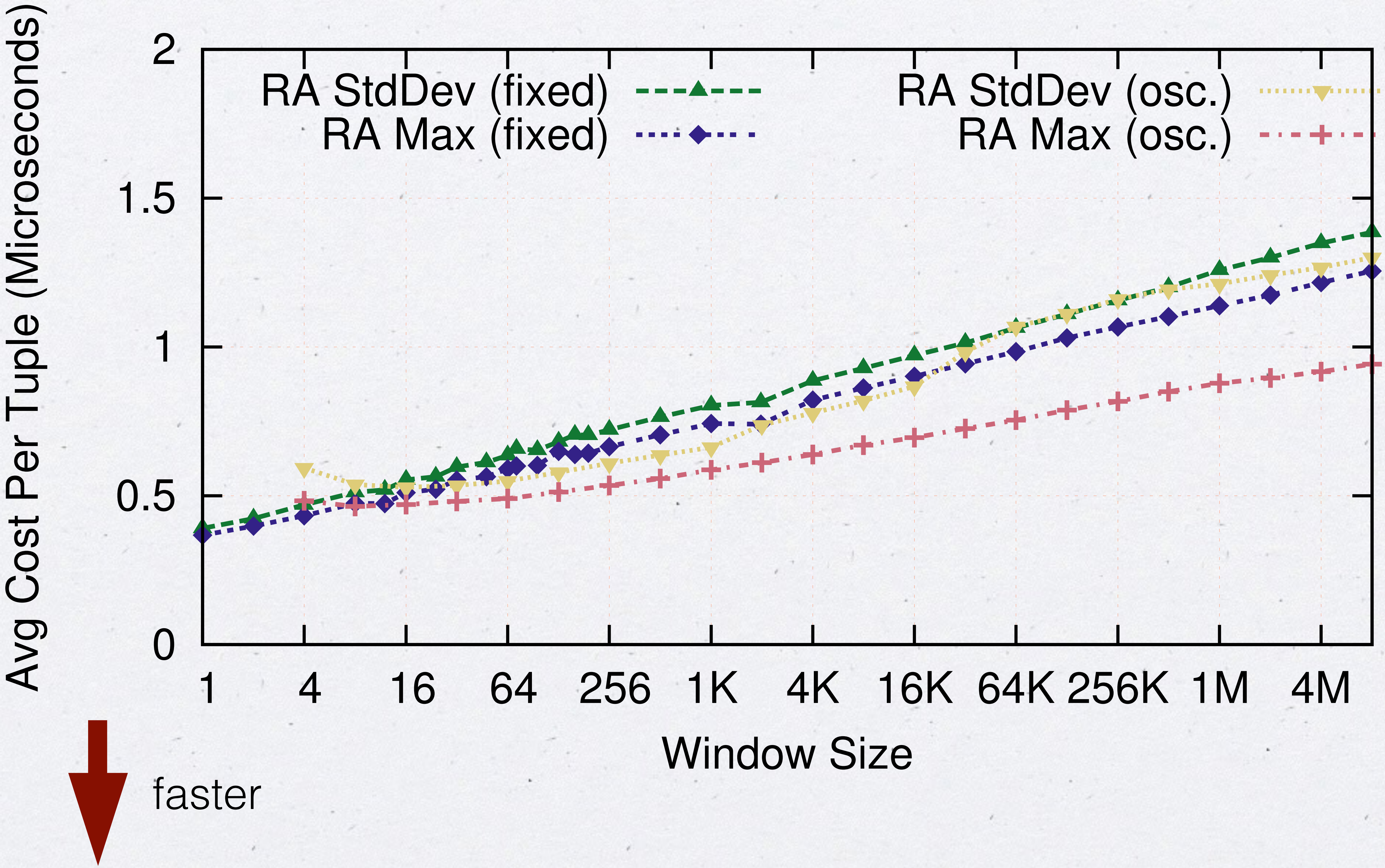
2

What's the performance trend under updates of different sizes?



3

How does wildly-changing window size affect the overall throughput?



Take-Home Points

This work: sliding-window aggregation

- easily extendible by users and has good performance
- careful systems + algorithmic design and engineering (blend of known ideas)
- general (non-FIFO, only need associativity) and fast for large & small windows

If m updates have been made on a window of size n, use $O(m\log(n/m))$ time to derive aggregate.

More details: see paper/come to the poster session.