New algorithms for distributed sliding windows

Sutanu Gayen
Joint work with N. V. Vinodchandran
University of Nebraska-Lincoln
Lincoln, NE USA

June 18, 2018
Single stream (aka streaming / infinite window model)

Cost of algorithm: \( \langle s, t \rangle \)
- \( s = \text{max. space used} \)
- \( t = \text{time to process each item} \)
Sliding window of a single stream

Cost of algorithm: \( \langle s, t \rangle \)

- \( s = \text{max. space used} \)
- \( t = \text{time to process each item} \)
Distributed infinite window (diw)

Cost of algorithm: \(\langle c, s, t \rangle\)
- \(c = \text{total communication}\)
- \(s = \text{max. space used}\)
- \(t = \text{time to process each item}\)
Distributed sliding window (dsw)

Cost of algorithm: \( \langle c, s, t \rangle \)
- \( c = \text{max. total communication per } W \text{ updates} \)
- \( s = \text{maximum space used} \)
- \( t = \text{time to process each item} \)

Assumption: Each item comes with a unique (modulo \( W \)) timestamp
dsw: Previous work and contribution

Previous work
Counting bits
Arithmetic mean
Distinct items
Sampling
Heavy hitters
Quantiles

Contribution
Frequency moments
Clustering
## Notations

<table>
<thead>
<tr>
<th>symbol</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$</td>
<td>the number of distributed nodes</td>
</tr>
<tr>
<td>$N_j$</td>
<td>$j$-th distributed node</td>
</tr>
<tr>
<td>$W$</td>
<td>window size</td>
</tr>
</tbody>
</table>
Main ‘transfer’ theorem (informal)

A function $f$ that satisfies certain ‘smoothness’ property

$$f \text{ has a diw algorithm of cost } \langle c, s, t \rangle$$

$$\Downarrow$$

$f$ has a dsw algorithm of cost $\langle W^x c, W^{1-x} + s, t \rangle$, upto polylog factors, for any $0 \leq x \leq 1$
(\(\alpha, \beta\))-Smooth function \(f\), \(\beta \leq \alpha\)

Defined by Braverman and Ostrovsky [FOCS 2007]

- \(f\) is non-decreasing
- \(f\) is non-negative
- \(f(A)\) is at most \(\text{poly}(|A|)\)
- For all sets \(A, B, C\) (stream parts) satisfies a continuity property:

\[
(1 - \beta)f(A \cup B) \leq f(B) \implies (1 - \alpha)f(A \cup B \cup C) \leq f(B \cup C)
\]
Example: Addition of bits is \((\epsilon, \epsilon)\)-smooth function

- Let \(f\) be the addition function of bits
- Consider the stream of bits: \[110100, 011010010011\]
- \(f(A) = 3, f(B) = 6, f(A \cup B) = 9\)
- \(f(A \cup B)/f(B) = 9/6\)
- Consider any \(C\) that has exactly \(x\) ones (e.g. \(0001000\))
- \(f(A \cup B \cup C)/f(B \cup C) = (9 + x)/(6 + x)\) (eg. 10/7)
- Above ratio never goes above 9/6
- In general, \(\leq \epsilon\) ratio now, ensures \(\leq \epsilon\) in future
General algorithm
Simple combination of local outputs does not work

\( F_0: \)
- Suppose each distributed node has seen only 5 of the \( n \) items: \{1, 2, 3, 4, 5\}
- Simple combination gives \( 5K \)
- Correct answer is 5

Clustering:
- Similarly, combination of local clusterings results in \( O(K) \) overall approximation ratio

We start with the ‘smooth histogram’ framework of Braverman and Ostrovsky [FOCS 2007] designed for computing certain functions over a sliding window of a single stream
‘Smooth histogram’ framework of Braverman and Ostrovsky [FOCS 2007]

- Defined to compute $f$ over a sliding window of a single stream
- Takes a single stream algorithm $\mathcal{A}$ for $f$ as a blackbox
- The framework gives a sliding window algorithm $\mathcal{B}$ that uses $\mathcal{A}$
- We give a similar framework for distributed streams
Smooth histogram data structure
For computing smooth functions over *single stream* sliding window

Properties of smooth histogram:

- A streaming algorithm is run from each index
  - values: $f_0, f_1, \ldots, f_i, \ldots$
- $l_0$ is expired and $l_1$ is active
- $(1 - \alpha)f_i < f_{i+1} < f_i$ but $f_{i+2} < (1 - \beta)f_i$, $\beta \leq \alpha$
  - At next index $f$ drops little
  - At next to next index $f$ drops large
Smooth histogram data structure
For computing smooth functions over *single stream* sliding window

Algorithm for maintaining smooth histogram:
Upon arrival of a new item \( x \)

- Start an index at \( x \)
- Remove redundant indices
  - For each index \( l_i \), find farthest index \( l_j \), such that \( f_j > (1 - \beta)f_i \)
  - Remove any index between \( i \) and \( j \)
  - Implies, at any later time \( f_j > (1 - \alpha)f_i \)
New algorithms for distributed sliding windows

- General algorithm
- Transfer theorem

Smooth histogram over dsw

- Smooth histogram must be maintained at the coordinator
- Previous algorithm starts an index at the arrival of each item
  - Coordinator is not directly aware of arrival
  - Communication cost $\Theta(W)$
New algorithms for distributed sliding windows

General algorithm

Transfer theorem

Smooth histogram over dsw: amortization

Static windows: \([1, W], \ldots, [aW + 1, aW + W], \ldots\)

At time \((aW + W)\) we construct the smooth histogram

Usable up to next \(W\) arrivals
New algorithms for distributed sliding windows

- General algorithm
- Transfer theorem

Smooth histogram over dsw

Construction at \((aW + W)\) for symmetric smooth functions: \(F_p\), clustering

- \(N_j\) saves the subset of window which appeared at \(N_j\)
  - Costs \(\Theta(W)\) space
- Two indices are made at \(aW + 1\) and at \((aW + W)\)
- A diw algorithm \(A\) for \(f\) is run backwards
  - \(A\) increases by \(\frac{1}{1-\beta}\) factor, an index is created
- Amortized time complexity: \(W \cdot t_A/W = t_A\)
New algorithms for distributed sliding windows

Smooth histogram over dsw
Construction at \((aW + W)\) for asymmetric smooth functions

- Two indices are made at \(aW + 1\) and at \((aW + W)\)
- \(I_{j+1}\) is created inductively based on \(I_j\)
  - Binary search for farthest index where \(f_j\) drops by \(< (1 - \beta)\) factor
- Communication and time complexity increases by polylog. factor
Main ‘transfer’ theorem (informal)

For $x = 0$

A function $f$ that satisfies certain ‘smoothness’ property

$$+$$

$f$ has a diw algorithm of cost $\langle c, s, t \rangle$

$$\Downarrow$$

$f$ has a dsw algorithm of cost $\langle W^x c, W^{1-x} + s, t \rangle$, upto polylog factors
New algorithms for distributed sliding windows

- General algorithm
- Transfer theorem

Smooth histogram over dsw
Space-communication trade off

- Split window into $W^x$ sub-windows each of size $W^{1-x}$
- Given two consecutive histograms, find their union histogram
- Remove redundant indices ($2L \rightarrow L$)
  - For each index $l_i$, find farthest index $l_j$, such that $f_j > (1 - \beta)f_i$
  - Remove any index between $i$ and $j$
  - Implies, at any later time $f_j > (1 - \alpha)f_i$
Smooth histogram over dsw

Space-communication trade off

- Split window into $W^x$ sub-windows each of size $W^{1-x}$
- Given two consecutive histograms, find their union histogram
- Remove redundant indices ($2L \rightarrow L$)
  - For each index $l_i$, find farthest index $l_j$, such that $f_j > (1 - \beta)f_i$
  - Remove any index between $i$ and $j$
  - Implies, at any later time $f_j > (1 - \alpha)f_i$
Main ‘transfer’ theorem (informal)

For any $0 \leq x \leq 1$

A function $f$ that satisfies certain ‘smoothness’ property

\[ + \]

$f$ has a diw algorithm of cost $\langle c, s, t \rangle$

\[ \Downarrow \]

$f$ has a dsw algorithm of cost $\langle W^x c, W^{1-x} + s, t \rangle$, upto polylog factors
Applications: frequency moments
New algorithms for distributed sliding windows

Applications: frequency moments

\(F_p, p\)-th frequency moment

- Each item of the data stream is an item from the universe \(\{1, 2, \ldots, n\}\)
- \(f_i\) be the number of occurrences of item \(i\) so far
- \(F_p = (f_1)^p + (f_2)^p + \cdots + (f_n)^p\)
- Example stream: \(\{2, 4, 4, 5, 4, 7, 8, 2, 2\}\)
  - Frequencies of \(\langle 2, 4, 5, 7, 8 \rangle\) are \(\langle 3, 3, 1, 1, 1 \rangle\)
  - \(F_0 = 5\)
  - \(F_2 = 3^2 + 3^2 + 1 + 1 + 1 = 21\)
- An important statistic in data analysis
- Approximation ratio \((1 \pm \epsilon)\): \((1 - \epsilon)F_p \leq \tilde{F}_p \leq (1 + \epsilon)F_p\)
New algorithms for distributed sliding windows

Applications: frequency moments

Frequency moments

\[ F_p \text{ satisfies } (\epsilon, \epsilon^p / p^p)\text{-'smoothness' property} \]

\[ F_p \text{ has a randomized diw algorithm of cost } \langle \frac{K^{p-1}}{\epsilon^{\Theta(p)}}, \frac{nK}{\epsilon}, \frac{1}{\epsilon^2} \rangle \]

\[ \Downarrow \]

\[ F_p \text{ has a dsw algorithm of cost } \langle W^x \frac{K^{p-1}}{\epsilon^{\Theta(p^2)}}, W^{1-x} + \frac{nK}{\epsilon^{\Theta(p)}}, \frac{1}{\epsilon^{\Theta(p)}} \rangle, \]

upto polylog factors, for any \( 0 \leq x \leq 1 \)

- For \( F_2 \), \( \langle W^x (\frac{K^2}{\epsilon^4} + \frac{K^{1.5}}{\epsilon^8}), W^{1-x} + \frac{K}{\epsilon^6}, \frac{1}{\epsilon^6} \rangle \)-cost dsw algorithm
Applications: Clustering
New algorithms for distributed sliding windows

Applications: Clustering

$k$-median clustering

- Each item of the data stream is a point in a metric space $\chi$
- Output $k$ points (medians) from the metric space to minimize sum over distances from each point to it’s nearest median
- $C^* = \arg \min_{C \subseteq \chi, |C| \leq k} \sum_{p \in P} \min_{c \in C} d(p, c)$ and $\text{OPT}_k = \sum_{p \in P} \min_{c \in C^*} d(p, c)$ where $d$ is the distance function of $\chi$
- Approximation ratio $r$: A set of medians $\tilde{C}^*$ with cost in $[\text{OPT}, r \cdot \text{OPT}]$
$k$-center clustering

- Each item of the data stream is a point in a metric space $\chi$.
- Output $k$ points (centers) from the metric space to minimize maximum over distances from each point to its nearest median.
- $C^* = \text{arg min}_{C \subseteq \chi, |C| \leq k} \max_{p \in P} \min_{c \in C} d(p, c)$ and $\text{OPT}_k = \max_{p \in P} \min_{c \in C^*} d(p, c)$ where $d$ is the distance function of $\chi$.
- Approximation ratio $r$: A set of medians $\tilde{C}^*$ with cost in $[\text{OPT}, r \cdot \text{OPT}]$. 

Applications: Clustering
Clustering costs are not smooth
Counterexample for $k$-median and $k$-center

- $OPT_k(A \cup B)$ and $OPT_k(B)$ are close
- But $OPT_k(A \cup B \cup C) >> OPT_k(B \cup C)$
Additional condition to make clustering smooth
Braverman, Lang, Levin and Monemizadeh [SODA 2016], for $k$-median clustering

- We would like to ensure: $\text{OPT}(A \cup B) \leq \gamma \text{OPT}(B)$ to get $\text{OPT}(A \cup B \cup C) \leq f(\gamma) \text{OPT}(B \cup C)$
- The following additional property suffices:
  - There is a $k$-median clustering $t : (A \cup B) \rightarrow [k]$ where each median $j$ satisfies $|t^{-1}(j) \cap A| \leq |t^{-1}(j) \cap B|$
We would like to ensure: \( \text{OPT}(A \cup B) \leq \gamma \text{OPT}(B) \) to get
\[ \text{OPT}(A \cup B \cup C) \leq f(\gamma) \text{OPT}(B \cup C) \]

The following additional property suffices:

There is a \( k \)-center clustering \( t : (A \cup B) \to [k] \) where each median \( j \) satisfies
\[ |t^{-1}(j) \cap A| > 0 \implies |t^{-1}(j) \cap B| > 0 \]
New algorithms for distributed sliding windows

Applications: Clustering

dsw algorithm for clustering

- Clustering costs are not exactly smooth
- We ensure an additional condition to ensure smoothness
- Gives $\langle k^2K, k^2K + W, k^2 \rangle$ cost algorithm, up to polylog factors
- We could not get a trade-off result
Future directions

- Trade-off results for clustering
- Sequence based dsw (no time-stamp provided)
  - Sampling is known