Tracking Distributed Aggregates over Time-based Sliding Windows

Graham Cormode

AT&T Labs

Ke Yi

HKUST

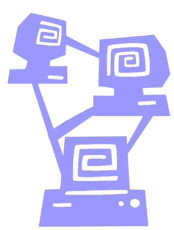
Distributed Monitoring

There are many scenarios where we need to track events:

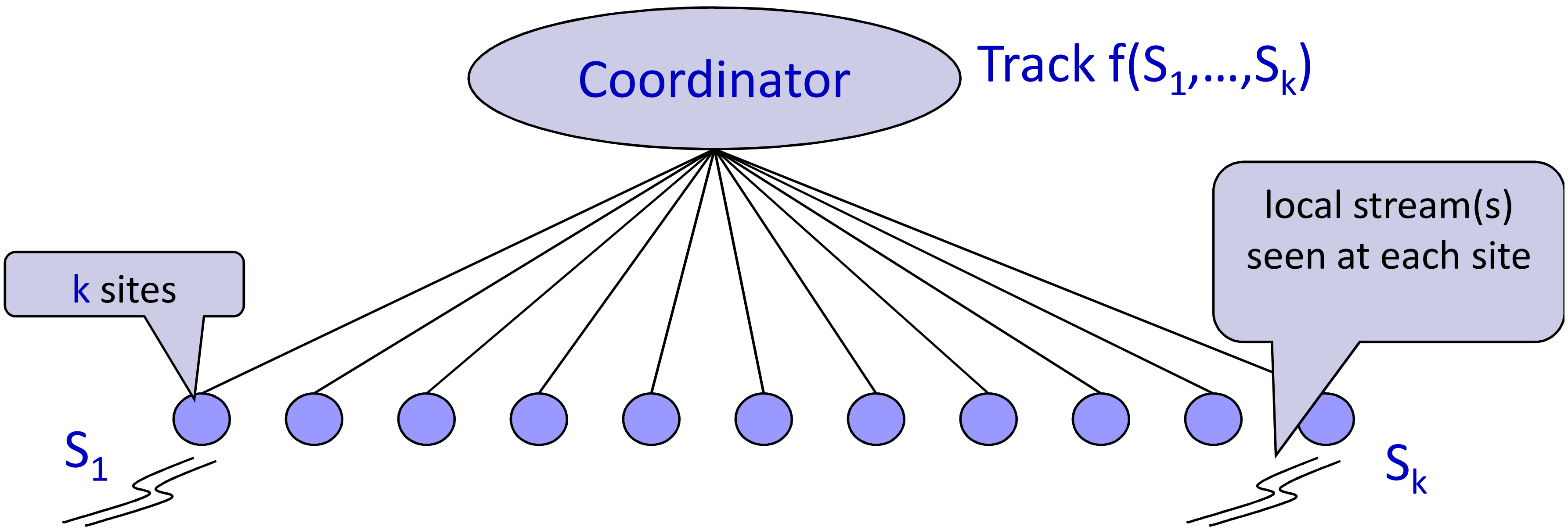
- Network health monitoring within a large ISP
- Collecting and monitoring environmental data with sensors
- Observing usage and abuse of distributed data centers

All can be abstracted as a collection of observers who want to collaborate to compute a function of their observations

From this we generate the Continuous Distributed Model



Continuous Distributed Model

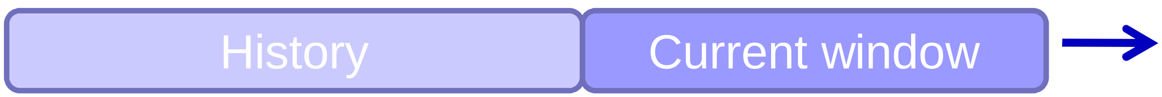


- Other structures possible (e.g., hierarchical)
- Site-site communication only changes things by factor 2
- **Goal**: Coordinator *continuously tracks* (global) function of streams
  - Achieve communication and space poly(k,1/ε,log n)
  - n events at a local site, adding up to N events globally

Prior Work in Distributed Monitoring

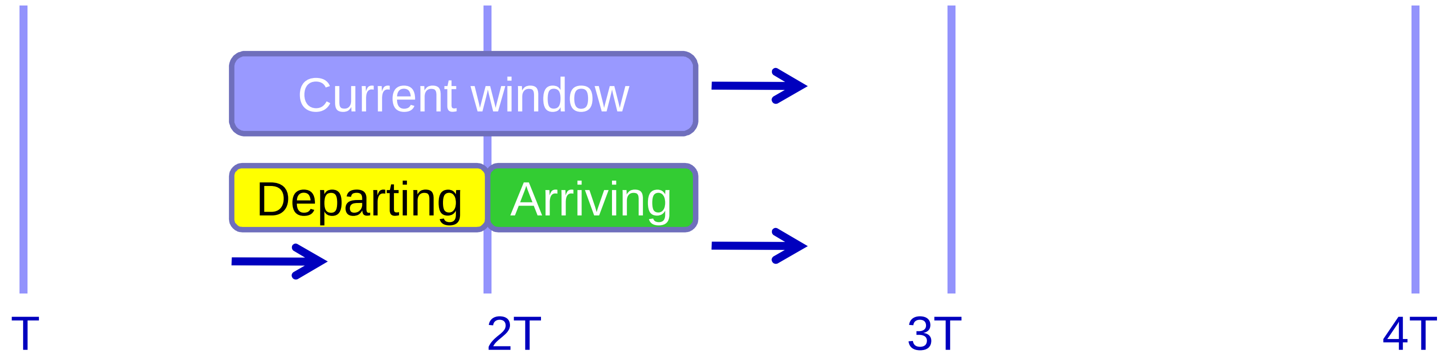
- Much interest in these problems in DB and TCS areas
- Track holistic functions of the (global) data distribution
  - Quantiles and heavy hitters [C, Garofalakis, Muthukrishnan, Rastogi 05]
  - Empirical Entropy [Arackaparambil Brody Chakrabarti 09]
  - Frequency Moments [C, Muthukrishnan, Yi 08]
  - Geometric approach [Sharman, Schuster, Keren 06]
- Track functions only over sliding window of recent events
  - Samples [C, Muthukrishnan, Yi, Zhang 10]
  - Counts and frequencies [Chan Lam Lee Ting 10]
- This work: new framework for monitoring over sliding windows

Sliding Window



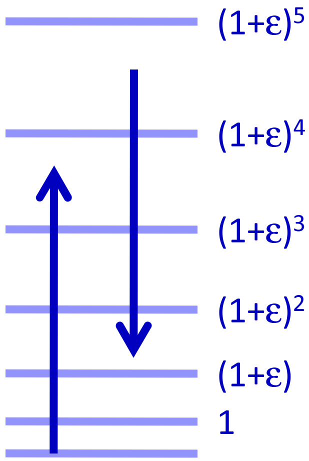
- In many cases, only care about recent events
- Prompts the sliding window model:
  - Only track events occurring within time T
- Consider several monitoring problems in sliding window:
  - Counting: (approximately) how many events in the window?
  - Heavy hitters: what are (approximate) heavy items in the window?
  - Quantiles: (approximate) the frequency distribution in the window

Forward/backward framework



- Key insight:
  - Complexity of sliding window comes from non-monotonicity
  - Break any window into forward (arrivals) and backward (expiries)
  - Solve each separately, improving overall
- Optimal results for several problems follow:
  - Counting: $O(k/\varepsilon \log(\varepsilon N/k))$ communication, $O(1/\varepsilon \log \varepsilon N)$ space
  - Heavy hitters: $O(k/\varepsilon \log(\varepsilon N/k))$ communication, $O(1/\varepsilon \log \varepsilon N)$ space
  - Quantiles: $O(k/\varepsilon \log^2 1/\varepsilon \log(\varepsilon N/k))$ comm, $O(1/\varepsilon \log^2 1/\varepsilon \log \varepsilon N)$ space

Warm up: Counting



- Forwards (for each site independently):
  - Within each (fixed) window, start a fresh counter
  - Update every time count increases by $(1+\varepsilon)$ factor
- Backwards (for each site independently):
  - Assume can keep all items from the last window
  - Send a message every time count decreases by $(1+\varepsilon)$ factor
- Analysis: $O(1/\varepsilon \log \varepsilon n)$ messages to count n items
  - Adds up to at most $O(k/\varepsilon \log(\varepsilon N/k))$ from all sites
- Make space efficient: keep "exponential histogram"
  - Takes space $O(1/\varepsilon \log \varepsilon n)$ space, reports $1+\varepsilon$ approx counts

Full Space Heavy Hitters Protocol

- Forward protocol broken into phases: $n \geq 2^a/\varepsilon$
  - Track counts of items as they arrive
  - Inform coordinator when a count goes up by 2^a
  - Ensures that coordinator knows counts accurate up to εn
- Backward protocol broken into phases: $n \leq 2^{a+1}/\varepsilon$
  - Inform coordinator of all counts more than 2^a at start of phase
  - Also inform when a count goes down by 2^a
  - Essentially reverse the forward protocol
- In both cases, at most $O(1/\varepsilon)$ communication per phase
- So $O(1/\varepsilon \log \varepsilon n)$ per site, $O(k/\varepsilon \log (\varepsilon N/k))$ total

Reduced Space Heavy Hitters

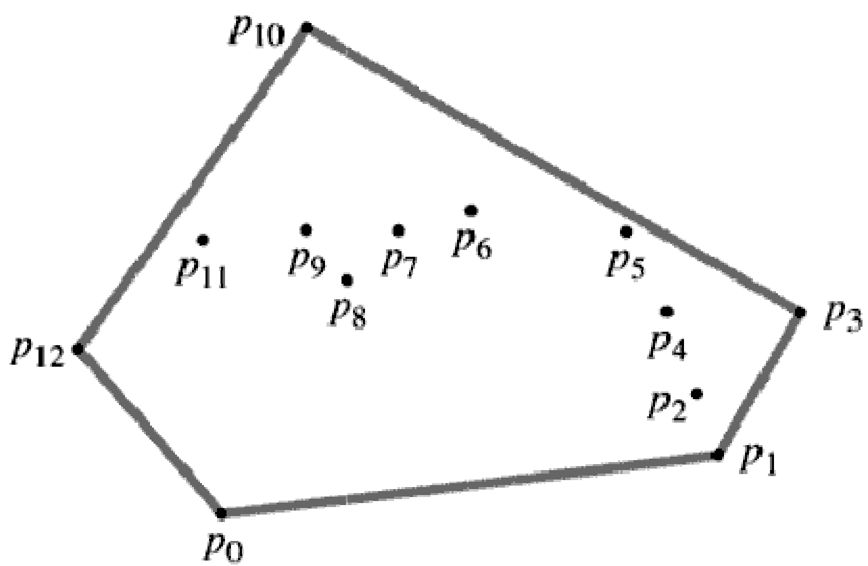
- Reduce space used by keeping only approximate information
  - Forward case: run a standard heavy hitters algorithm
  - Keep $O(1/\varepsilon)$ items and counts, and obtain εn accuracy
  - Backward case: run a sliding-window heavy hitters algorithm
  - Keep $O(1/\varepsilon)$ items, counts & timestamps, get $\varepsilon 2^a$ accuracy
  - Total space reduced to $O(1/\varepsilon \log n)$ per site
- Coordinator space $O(k/\varepsilon)$: keep current heavy hitters from each site
- Communication remains $O(k/\varepsilon \log (\varepsilon N/k))$ per window

Quantiles

- Forward protocol: guess window sizes $W = 2^a/\varepsilon$
  - For each W, further break window down into blocks
  - Keep a compact quantile summary of each block
  - Send summary to coordinator when a block fills
  - Any window can be broken into a few blocks
  - Tolerate a little imprecision in block size within error bounds
- Backward protocol: almost identical to forward
  - Just need to keep track of recent blocks locally
  - Only send needed summaries to coordinator at end of window
- Communication cost $O(k/\varepsilon \log^2 (1/\varepsilon) \log (N/k))$ per window
- Space $O(1/\varepsilon \log^2 (1/\varepsilon) \log \varepsilon n)$ at each site

Other Functions

- Can use similar approach for several other functions:
  - Distinct counts: track unique items seen across sites
  - Entropy: track the entropy of the observed frequency distribution
  - Geometric functions: diameter and convex hull of points



Concluding Remarks

- Introduced forward/backward framework for monitoring
  - Allows efficient solutions for sliding window problems
  - Improves on bounds from prior work [Chan et al 2010]
  - Simplifies analysis, reduces cases to handle
  - (Near) optimal solutions for counting and heavy hitters
- Open problems:
  - Bounds for quantiles are not optimal by $\log(1/\varepsilon)$ factors
  - Extend to other problems?
  - Build into systems, empirical studies