Sketching Streams through the Net: Distributed Approximate Query Tracking

Minos Garofalakis
Intel Research Berkeley
minos.garofalakis@intel.com

(Joint work with Graham Cormode, Bell Labs)

Continuous Distributed Queries

Traditional data management supports one shot queries
- May be look-ups or sophisticated data management tasks, but tend to be on-demand
- New large scale data monitoring tasks pose novel data management challenges

Continuous, Distributed, High Speed, High Volume...
Network Monitoring Example

Network Operations Center (NOC) of a major ISP
- Monitoring 100s of routers, 1000s of links and interfaces, millions of events / second
- Monitor all layers in network hierarchy (physical properties of fiber, router packet forwarding, VPN tunnels, etc.)

Other applications: distributed data centers/web caches, sensor networks, power grid monitoring, ...

Common Aspects / Challenges

Monitoring is Continuous...
- Need real-time tracking, not one-shot query/response

...Distributed...
- Many remote sites, connected over a network, each sees only part of the data stream(s)
- Communication constraints

...Streaming...
- Each site sees a high speed stream of data, and may be resource (CPU/Memory) constrained

...Holistic...
- Track quantity/query over the global data distribution

...General Purpose...
- Can handle a broad range of queries
Each stream distributed across a (sub)set of remote sites
- E.g., stream of UDP packets through edge routers

**Challenge:** Continuously track holistic query at coordinator
- More difficult than single-site streams
- Need space/time *and communication* efficient solutions

**But...** exact answers are not needed
- Approximations with accuracy guarantees suffice
- Allows a tradeoff between accuracy and communication/processing cost

---

**Prior Work – Specialized Solutions**

<table>
<thead>
<tr>
<th></th>
<th>continuous</th>
<th>distributed</th>
<th>streaming</th>
<th>holistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed top-k &amp; quantiles</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Streaming top-k &amp; quantiles</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Distributed top-k</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Distributed filters</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
</tbody>
</table>

*First general-purpose approach for broad range of distributed queries*
**System Architecture**

Streams at each site add to (or, subtract from) multisets/frequency distribution vectors $f_i$

- More generally, can have hierarchical structure

**Queries**

“Generalized” inner-products on the $f_i$ distributions

$$|f_i \otimes f_j| = f_i \cdot f_j = \sum_v f_i[v]f_j[v]$$

Capture join/multi-join aggregates, range queries, heavy hitters, approximate histograms/wavelets, ...

Allow approximation: Track $f_i \cdot f_j \pm \varepsilon \parallel f_i \parallel \parallel f_j \parallel$

Goal: Minimize communication/computation overhead

- Zero communication if data distributions are “stable”
Our Solution: An Overview

- General approach: "In-Network" Processing
  - Remote sites monitor local streams, tracking deviation of local distribution from predicted distribution
  - Contact coordinator only if local constraints are violated

- Use concise sketch summaries to communicate...
  Much smaller cost than sending exact distributions

- No/little global information
  Sites only use local information, avoid broadcasts

- Stability through prediction
  If behavior is as predicted, no communication

AGMS Sketching 101

Goal: Build small-space summary for distribution vector \( f[v] \ (v=1,\ldots, N) \) seen as a stream of \( v \)-values

Data stream: \[ 3, 1, 2, 4, 2, 3, 5, \ldots \] → 

Basic Construct: Randomized Linear Projection of \( f \) = project onto dot product of \( f \)-vector

\[
X = \sum_v f[v] \xi_v \quad \text{where} \quad \xi = \text{vector of random values from an appropriate distribution}
\]

- Simple to compute: Add \( \xi_v \) whenever the value \( v \) is seen

Data stream: \[ 3, 1, 2, 4, 2, 3, 5, \ldots \] → \[
\xi_1 + 2\xi_2 + 2\xi_3 + \xi_4 + \xi_5
\]

- Generate \( \xi_v \)'s in small (logN) space using pseudo-random generators

Intel Research
AGMS Sketching 101 (contd.)

$$f \begin{array}{cccc} 1 & 2 & 2 & 1 \\ \{\xi_v\} \end{array} \quad \begin{array} { c c c c } 1 & \sum_v f[v] \xi_v = \\ \cdot \xi_1 + 2\xi_2 + 2\xi_3 + \xi_4 + \xi_5 \\ \cdot \\ \cdot \end{array} \quad \begin{array} { c c c c } X_i = \sum_v f[v] \psi_v \\ X_m = \sum_v f[v] \psi_v \\ \end{array}$$

Simple randomized linear projections of data distribution

- Easily computed over stream using logarithmic space
- Linear: Compose through simple addition

Theorem[AGMS]: Given sketches of size \(O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)\)

\[\text{sk}(f_i) \cdot \text{sk}(f_j) \in f_i \cdot f_j \pm \epsilon \| f_i \| f_j \|\]

Sketch Prediction

Sites use AGMS sketches to summarize local streams

- Compose to sketch the global stream \(\text{sk}(f_i) = \sum_s \text{sk}(f_{is})\)
- BUT... cannot afford to update on every arrival!

Key idea: Sketch prediction

- Try to predict how local-stream distributions (and their sketches) will evolve over time
- Concise sketch-prediction models, built locally at remote sites and communicated to coordinator

- Shared knowledge on expected local-stream behavior over time
- Allow us to achieve stability
Sketch Prediction (contd.)

\[ f^{p}_{is} \]

Predicted Distribution

\[ sk^{p}(f_{is}) \]

Predicted Sketch

\[ f_{is} \]

True Distribution (at site)

\[ sk(f_{is}) \]

True Sketch (at site)

Prediction used at coordinator for query answering

Prediction error tracked locally by sites (local constraints)

Query Tracking Scheme

Overall error guarantee at coordinator is function \( g(\varepsilon, \theta) \)

- \( \varepsilon \) = local-sketch summarization error (at remote sites)
- \( \theta \) = upper bound on local-stream deviation from prediction
  - “Lag” between remote-site and coordinator view

Exact form of \( g(\varepsilon, \theta) \) depends on the specific query \( Q \) being tracked

BUT... local site constraints are the same

- L2-norm deviation of local sketches from prediction
**Query Tracking Scheme** *(contd.)*

Continuously track  

\[ Q = |f_i \otimes f_j| \]

**Remote Site protocol**

- Each site \( s \in \text{sites}(f_i) \) maintains \( \mathcal{E} \)-approx. sketch \( \text{sk}(f_{is}) \)
- On each update check L2 deviation of predicted sketch

\[
(*) \quad \|\text{sk}(f_{is}) - \text{sk}^p(f_{is})\| \leq \frac{\theta}{\sqrt{k_i}} \|\text{sk}(f_{is})\|
\]

- If (*) fails, send up-to-date sketch and (perhaps) prediction model info to coordinator

---

**Query Tracking Scheme** *(contd.)*

**Coordinator protocol**

- Use site updates to maintain sketch predictions \( \text{sk}^p(f_i) \)
- At any point in time, estimate

\[
|f_i \otimes f_j| \approx \text{sk}^p(f_i) \cdot \text{sk}^p(f_j)
\]

**Theorem:** If (*) holds at participating remote sites, then

\[
\text{sk}^p(f_i) \cdot \text{sk}^p(f_j) \in |f_i \otimes f_j| \pm (\mathcal{E} + 2\theta) \|f_i\| \|f_j\|
\]

**Extensions:** Multi-joins, wavelets/histograms, sliding windows, exponential decay, ...

**Key Insight:** Under (*), predicted sketches at coordinator are \( g(\mathcal{E}, \theta) \)-approximate
Sketch-Prediction Models

Simple, concise models of local-stream behavior
- Sent to coordinator to keep site/coordinate “in-sync”

Different Alternatives
- Static model: No change in distribution since last update
  - Naïve, “no change” assumption: $\mathbf{sk}^p(f(t)) = \mathbf{sk}(f(t_{\text{prev}}))$
  - No model info sent to coordinator

\[
\begin{align*}
  f(t_{\text{prev}}) & \quad \rightarrow \quad f^p(t) \\
\end{align*}
\]

Sketch-Prediction Models (contd.)
- Linear-growth model: Uniformly scale distribution by time ticks
  - $\mathbf{sk}^p(f(t)) = \frac{t}{t_{\text{prev}}} \mathbf{sk}(f(t_{\text{prev}}))$ (by sketch linearity)
  - Model “synchronous/uniform updates”
  - Again, no model info needed

\[
\begin{align*}
  f(t_{\text{prev}}) & \quad \rightarrow \quad f^p(t) = \frac{t}{t_{\text{prev}}} f(t_{\text{prev}}) \\
\end{align*}
\]
Sketch-Prediction Models (contd.)

- Velocity/acceleration model: Predict change through “velocity” & “acceleration” vectors from recent local history
  - Velocity model: \( f^p(t) = f(t_{prev}) + \Delta t \cdot \nu \)
    - Compute velocity vector over window of \( W \) most recent updates to stream
  - By sketch linearity \( \text{sk}^p(f(t)) = \text{sk}(f(t_{prev})) + \Delta t \cdot \text{sk}(\nu) \)
  - Just need to communicate one more sketch (for the velocity vector)!

\[
f(t_{prev}) \quad \xrightarrow{\text{Sketch}} \quad f^p(t) = f(t_{prev}) + \Delta t \cdot \nu
\]

Sketch-Prediction: Summary

<table>
<thead>
<tr>
<th>Model</th>
<th>Info</th>
<th>Predicted Sketch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>\emptyset</td>
<td>( \text{sk}^p(f(t)) = \text{sk}(f(t_{prev})) )</td>
</tr>
<tr>
<td>Linear growth</td>
<td>\emptyset</td>
<td>( \text{sk}^p(f(t)) = \frac{t}{t_{prev}} \text{sk}(f(t_{prev})) )</td>
</tr>
<tr>
<td>Velocity/Acceleration</td>
<td>\text{sk}(\nu)</td>
<td>( \text{sk}^p(f(t)) = \text{sk}(f(t_{prev})) + \Delta t \cdot \text{sk}(\nu) )</td>
</tr>
</tbody>
</table>

- Communication cost analysis: comparable to *one-shot* sketch computation
- Many other models possible – not the focus here...
  - Need to carefully balance power & conciseness
Improving Basic AGMS

Update time for basic AGMS sketch is $\Omega(|\text{sketch}|)$

BUT...

- Sketches can get large -- cannot afford to touch every counter for rapid-rate streams!
  - Complex queries, stringent error guarantees, ...
- Sketch size may not be the limiting factor (PCs with GBs of RAM)

The Fast AGMS Sketch

*Fast AGMS Sketch:* Organize the atomic AGMS counters into hash-table buckets

- Each update touches only a few counters (one per table)
- Same space/accuracy tradeoff as basic AGMS (in fact, slightly better©)
- BUT, *guaranteed logarithmic update times* (regardless of sketch size)!!
**Experimental Study**

Prototype implementation of query-tracking schemes in C

Measured improvement in communication cost (compared to sending all updates)

Ran on real-life data

- World Cup 1998 HTTP requests, 4 distributed sites, about 14m updates per day

Explored

- Accuracy tradeoffs ($\varepsilon$ vs. $\theta$)
- Effectiveness of prediction models
- Benefits of Fast AGMS sketch

---

**Accuracy Tradeoffs – V/A Model**

1 Day HTTP data, $W=20000$

- $\varepsilon + 2\theta = 10\%$
- $\varepsilon + 2\theta = 4\%$
- $\varepsilon + 2\theta = 2\%$

Large “sweetspot” for dividing overall error tolerance
**Prediction Models**

1 Day HTTP data, $\varepsilon = 2\theta$

- $\varepsilon = \theta = 5\%$
- $\varepsilon = \theta = 2\%$
- $\varepsilon = \theta = 1\%$

**Stability – V/A Model**

8 Days HTTP requests, $\varepsilon = 2\theta, W = 20000$

- $\varepsilon = \theta = 5\%$
- $\varepsilon = \theta = 2\%$
- $\varepsilon = \theta = 1\%$
**Fast AGMS vs. Standard AGMS**

![Graph showing comparison between 1 Day HTTP data, $\varepsilon=2\theta$, 14 million updates for Static, Static-Fast, Velocity/Acceleration, and Velocity/Acceleration-Fast.]

**Conclusions & Future Directions**

Novel algorithms for communication-efficient distributed approximate query tracking

- **Continuous, sketch-based** solution with error guarantees
- **General-purpose**: Covers a broad range of queries
- “In-network” processing using **simple, localized constraints**
- **Novel sketch structures** optimized for rapid streams

**Open problems**

- Specialized solutions optimized for specific query classes?
- More clever prediction models (e.g., capturing correlations across sites)?
- Efficient distributed trigger monitoring?
Accuracy – Total Error

1 Day HTTP data, \( \Theta = 5\% \), \( W = 20000 \)

- Error bound
- Static
- Velocity-Acceleration

Total Error in Self-Join

\[ \varepsilon \]
Accuracy – Tracking Error

1 Day HTTP data, ε=5%, W=20000

- Error bound
- Static
- Velocity-Acceleration

Tracking Error in Self-join

Other Monitoring Applications

Sensor networks
- Monitor habitat and environmental parameters
- Track many objects, intrusions, trend analysis...

Utility Companies
- Monitor power grid, customer usage patterns etc.
- Alerts and rapid response in case of problems