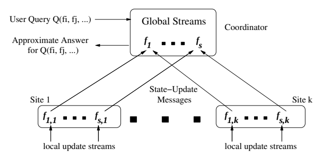
Sketching Streams through the Net: Distributed Approximate Query Tracking



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

(Joint work with Graham Cormode, Bell Labs)

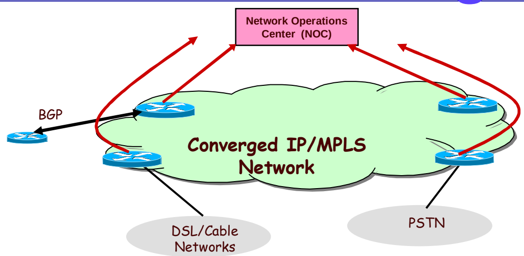
Continuous Distributed Queries

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, …

Intel **Research**

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…

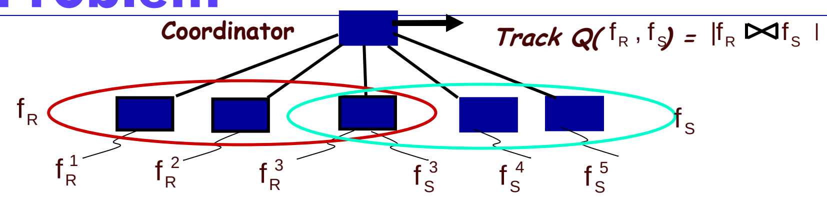
- Track quantity/query over the *global* data distribution

…General Purpose…

- Can handle a *broad range of queries*

Intel **Research**

Problem



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

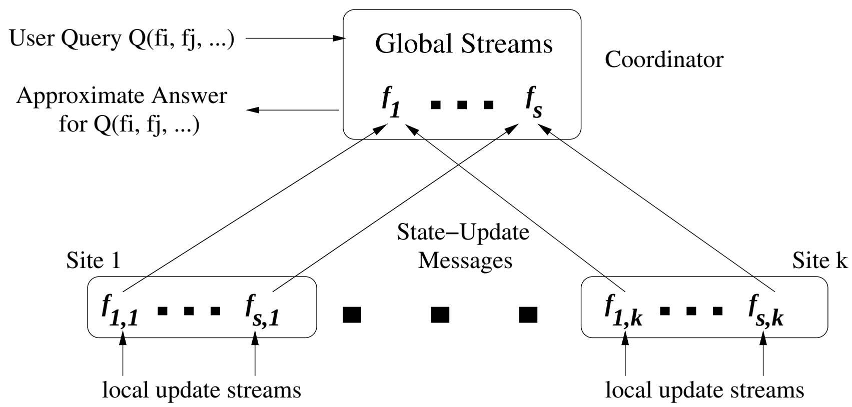
- Approximations with accuracy guarantees suffice
- Allows a tradeoff between accuracy and communication/ processing cost

Prior Work – Specialized Solutions

	continuous	distributed	streaming	holistic	
Distributed top-k & quantiles	X	✓	✓	✓	GK04, MSDO05
	✓	✓	✓	✓	*CGMR05*
Streaming top-k & quantiles	✓	X	✓	✓	GK01, MM02
Distributed top-k	✓	✓	X	✓	BO03
Distributed filters	✓	✓	✓	X	OJW03

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

Intel **Research**

Queries

"Generalized" inner-products on the f_i distributions

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

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

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

Goal: Minimize communication/computation overhead

–Zero communication if data distributions are "stable"

Intel **Research**

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

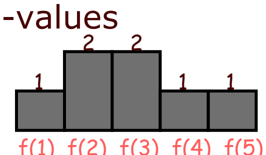
- Use concise sketch summaries to communicate… Much smaller cost than sending exact distributions
- No/little global information Sites only use local information, avoid broadcasts
- Stability through prediction If behavior is as predicted, no communication

Intel **Research**

AGMS Sketching 101

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

Data stream: 3, 1, 2, 4, 2, 3, 5, . . . →



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

$$X = \sum_v f[v]\xi_v$$ where ξ = vector of random values from an appropriate distribution

- Simple to compute: Add ξ_v whenever the value v is seen

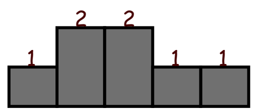
Data stream: 3, 1, 2, 4, 2, 3, 5, . . . → $\xi_1 + 2\xi_2 + 2\xi_3 + \xi_4 + \xi_5$

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

Intel **Research**

AGMS Sketching 101 *(contd.)*

f



$$\text{sk}(f) = \left[\begin{array}{l} X_1 = \sum_v f[v]\xi_v = \xi_1 + 2\xi_2 + 2\xi_3 + \xi_4 + \xi_5 \\ \vdots \\ X_m = \sum_v f[v]\psi_v \end{array}\right] \qquad \{\xi_v\},\ \{\psi_v\}$$

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 $O(\frac{\log(1/\delta)}{\varepsilon^2})$

$$\text{sk}(f_i) \cdot \text{sk}(f_j) \in f_i \cdot f_j \pm \varepsilon \| f_i \| \| f_j \|$$

Intel **Research**

Sketch Prediction

Sites use AGMS sketches to summarize local streams

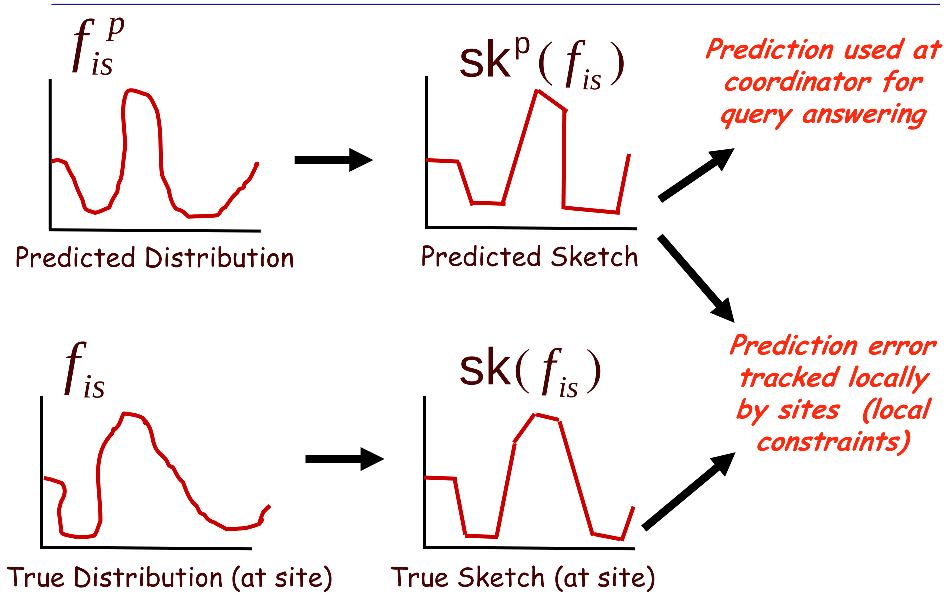
- Compose to sketch the global stream $\text{sk}(f_i) = \sum_s \text{sk}(f_{is})$
- BUT… cannot afford to update on every arrival!

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*

Intel **Research**

Sketch Prediction *(contd.)*



Intel **Research**

Query Tracking Scheme

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

- ε = local-sketch summarization error (at remote sites)
- θ = 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

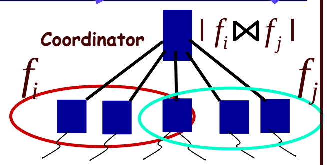
BUT… local site constraints are the same

- L2-norm deviation of local sketches from prediction

Intel **Research**

Query Tracking Scheme *(contd.)*

Continuously track Q = $|f_i \bowtie f_j|$



Remote Site protocol

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

(*) $$\|sk(f_{is}) - sk^p(f_{is})\| \le \frac{\theta}{\sqrt{k_i}} \|sk(f_{is})\|$$

- 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 $sk^p(f_i)$
- At any point in time, estimate

$$|f_i \bowtie f_j| \approx sk^p(f_i) \cdot sk^p(f_j)$$

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

$$sk^p(f_i) \cdot sk^p(f_j) \in |f_i \bowtie f_j| \pm (\varepsilon + 2\theta) \|f_i\| \|f_j\|$$

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

Key Insight: Under (*), predicted sketches at coordinator are $g(\varepsilon, \theta)$ -*approximate*

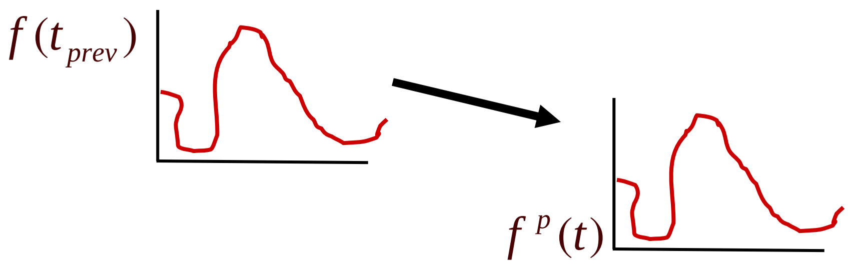
Sketch-Prediction Models

Simple, concise models of local-stream behavior

- Sent to coordinator to keep site/coordinator "in-sync"

Different Alternatives

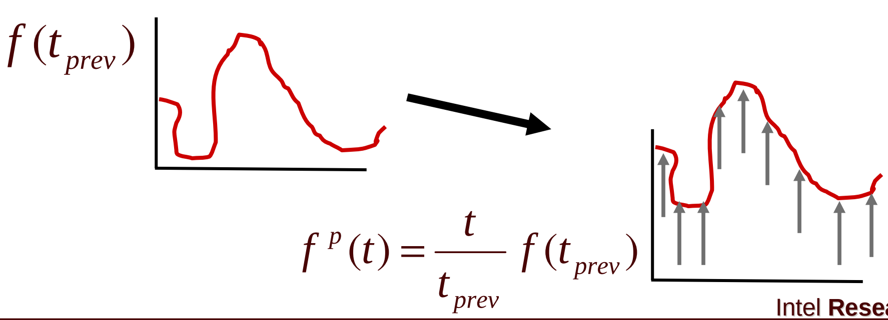
- Static model: No change in distribution since last update
  - Naïve, "no change" assumption: $\mathrm{sk}^{p}(f(t)) = \mathrm{sk}(f(t_{prev}))$
  - No model info sent to coordinator



Intel **Research**

Sketch-Prediction Models *(contd.)*

- Linear-growth model: Uniformly scale distribution by time ticks
  - $\mathrm{sk}^{p}(f(t)) = \frac{t}{t_{prev}} \mathrm{sk}(f(t_{prev}))$ (by sketch linearity)
  - Model "synchronous/uniform updates"
  - Again, no model info needed



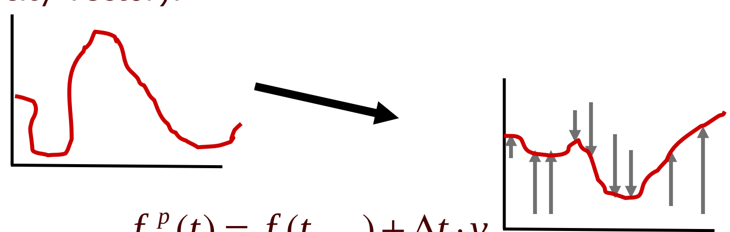
Intel **Research**

Sketch-Prediction Models *(contd.)*

– Velocity/acceleration model: Predict change through "velocity" & "acceleration" vectors from recent local history

- Velocity model: $f^{p}(t) = f(t_{prev}) + \Delta t \cdot v$
  - Compute velocity vector over window of W most recent updates to stream
- By sketch linearity $\mathrm{sk}^{p}(f(t)) = \mathrm{sk}(f(t_{prev})) + \Delta t \cdot \mathrm{sk}(v)$
- Just need to communicate one more sketch (for the velocity vector)!

$f(t_{prev})$



$f^{p}(t) = f(t_{prev}) + \Delta t \cdot v$

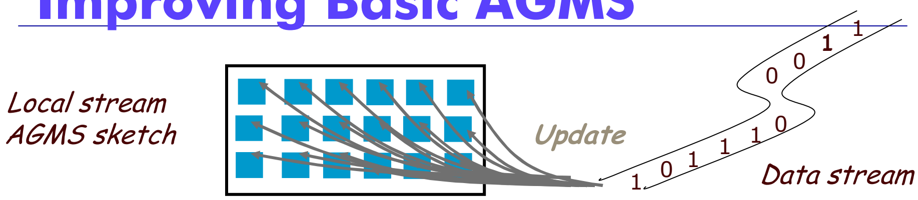
Sketch-Prediction: Summary

Model	Info	Predicted Sketch
Static	$\emptyset$	$\mathrm{sk}^{p}(f(t)) = \mathrm{sk}(f(t_{prev}))$
Linear growth	$\emptyset$	$\mathrm{sk}^{p}(f(t)) = \frac{t}{t_{prev}} \mathrm{sk}(f(t_{prev}))$
Velocity/ Acceleration	$\mathrm{sk}(v)$	$\mathrm{sk}^{p}(f(t)) = \mathrm{sk}(f(t_{prev})) + \Delta t \cdot \mathrm{sk}(v)$

- 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 $\Omega(|\text{sketch}|)$

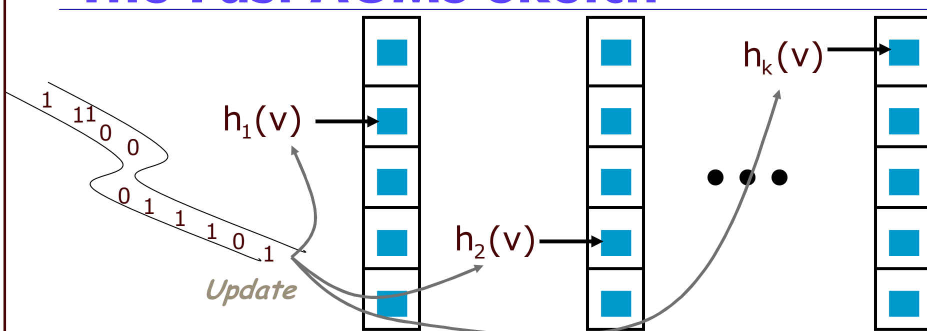
BUT…

–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

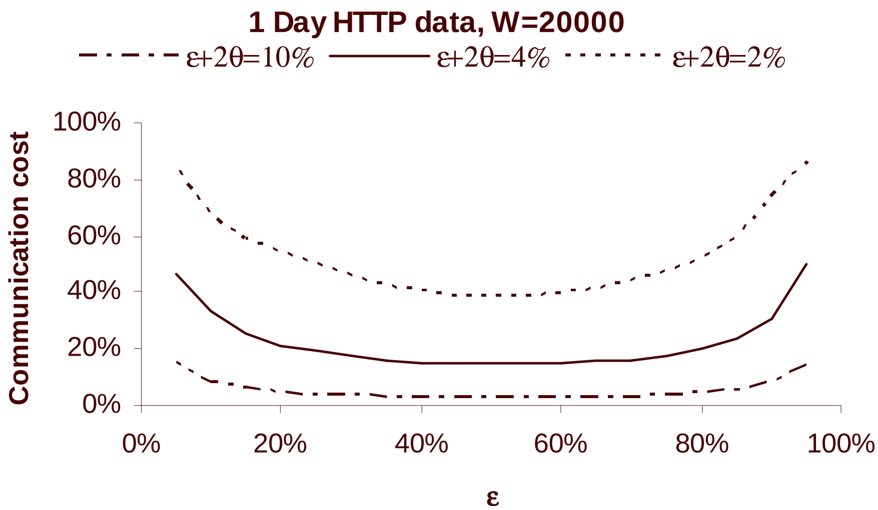
- World Cup 1998 HTTP requests, 4 distributed sites, about 14m updates per day

Explored

- Accuracy tradeoffs (ε vs. θ)
- Effectiveness of prediction models
- Benefits of Fast AGMS sketch

Intel **Research**

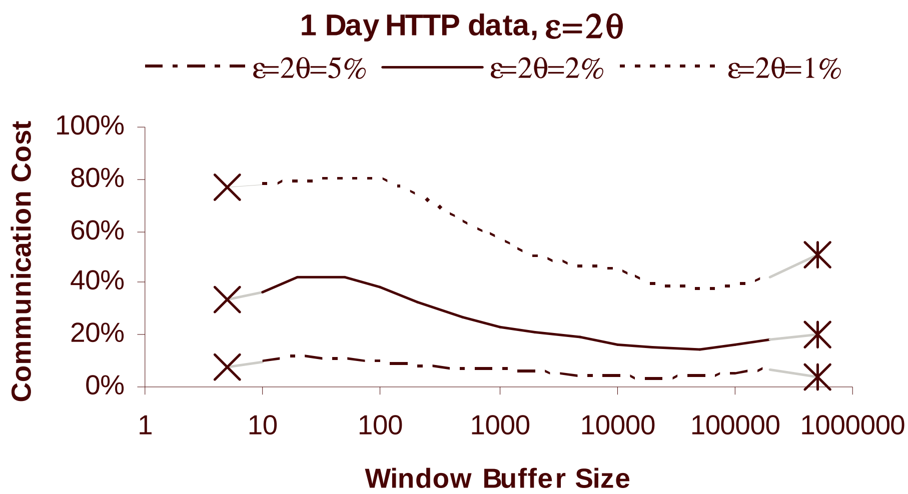
Accuracy Tradeoffs – V/A Model



Large "sweetspot" for dividing overall error tolerance

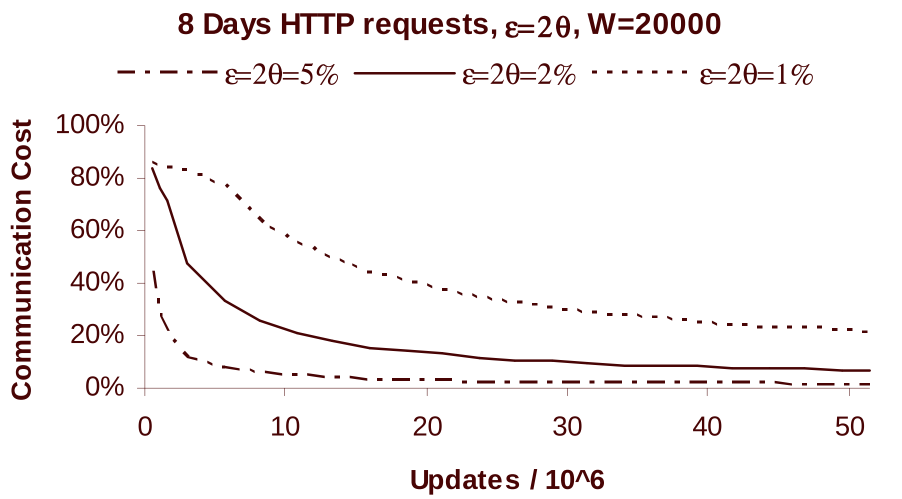
Intel **Research**

Prediction Models



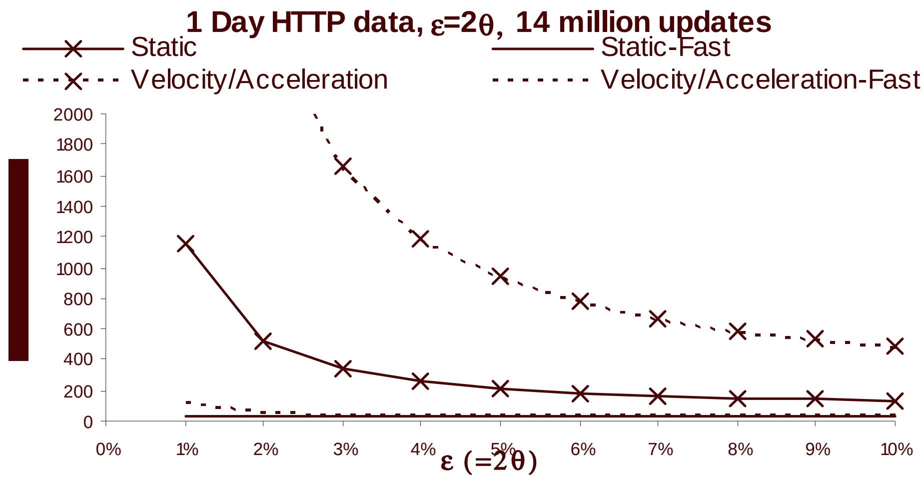
Intel **Research**

Stability – V/A Model



Intel **Research**

Fast AGMS vs. Standard AGMS



Conclusions & Future Directions

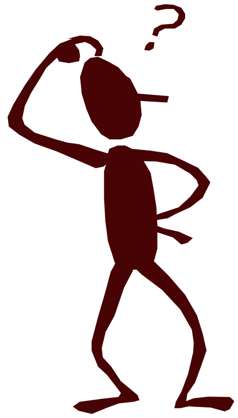
Novel algorithms for communication-efficient distributed approximate query tracking

- 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?

Thank you!

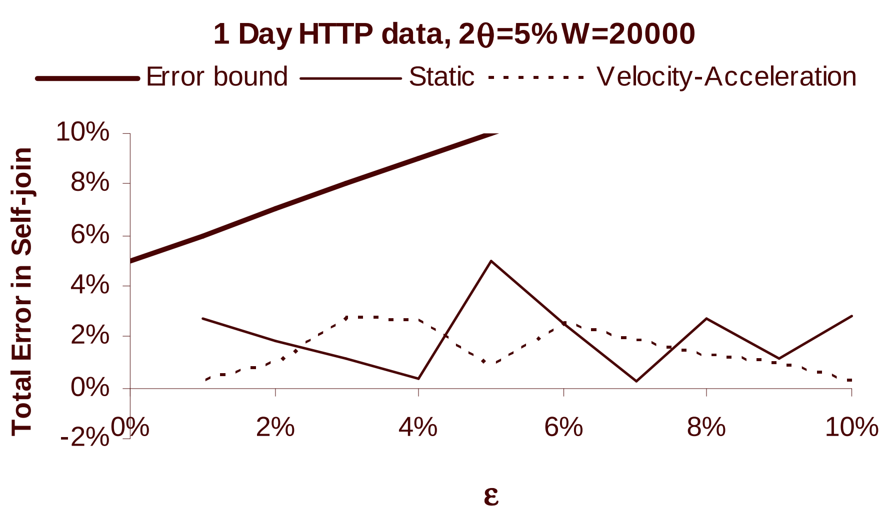


http://www2.berkeley.intel-research.net/~minos/

minos.garofalakis@intel.com

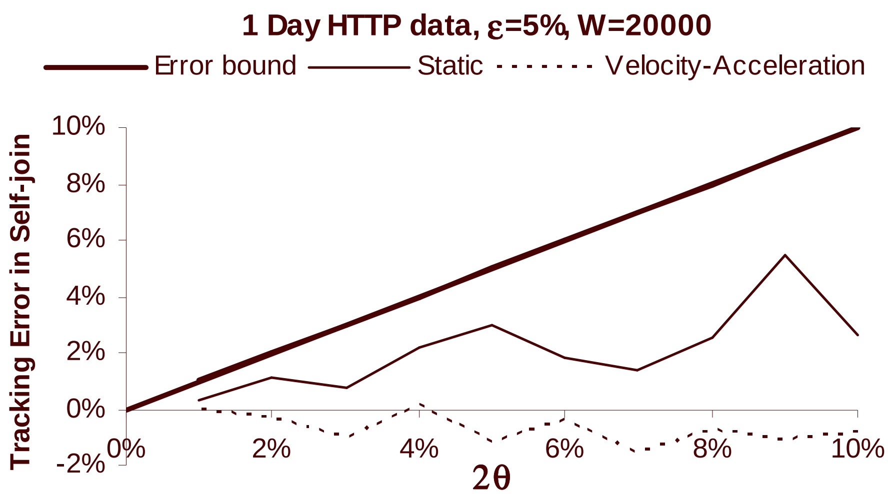
Intel **Research**

Accuracy – Total Error



Intel **Research**

Accuracy – Tracking Error



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