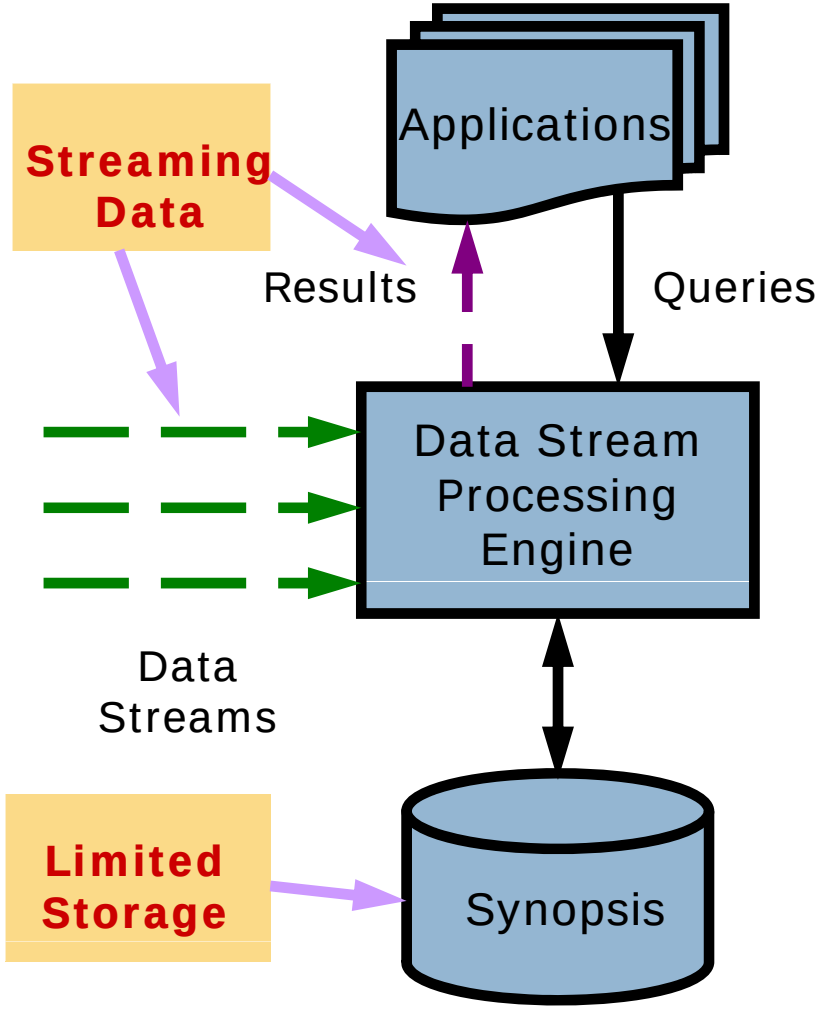
Data Stream and its Real Time Processing

Aoying Zhou (周傲英)
Institute of Massive computing
(**IMC**)
East China Normal University
(**ECNU**)

Outline

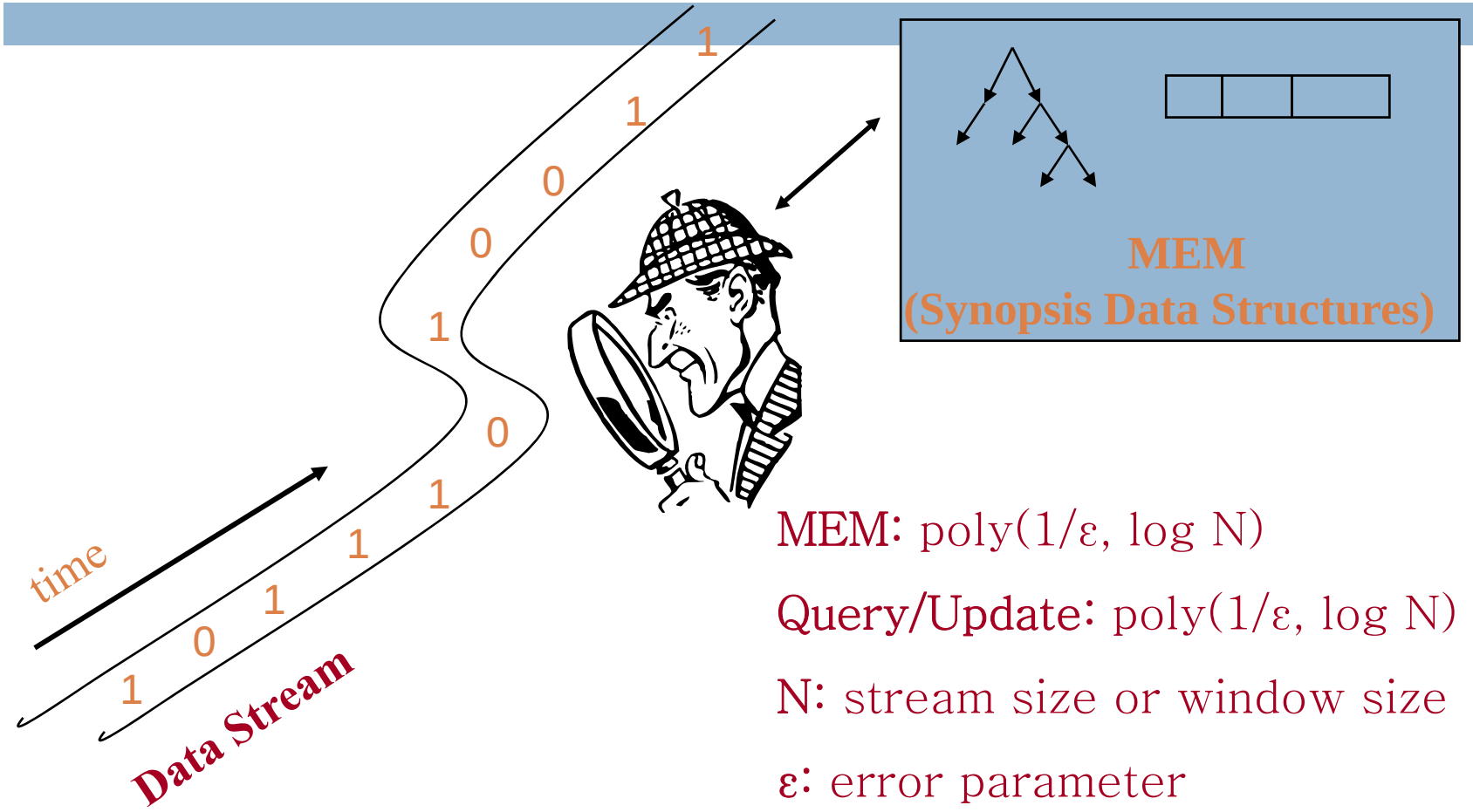
- Overview of data stream
- Some research work done by us
  - Frequent pattern
  - Clustering and density estimate
  - Streaming system
  - Aggregate query
  - Complex data
  - Uncertain stream
- Conclusions and future work

Data Stream and Queries



- Data streams
  - Digits (e.g., head of a data packet)
  - Structured tuple (e.g., Web-log)
  - XML documents (e.g., pub/sub system)
- Queries
  - Continuous/ad hoc Queries
  - May have different complexity
    - Counting
    - Structured queries (SQL)
    - Data analysis and mining

Streaming Algorithms



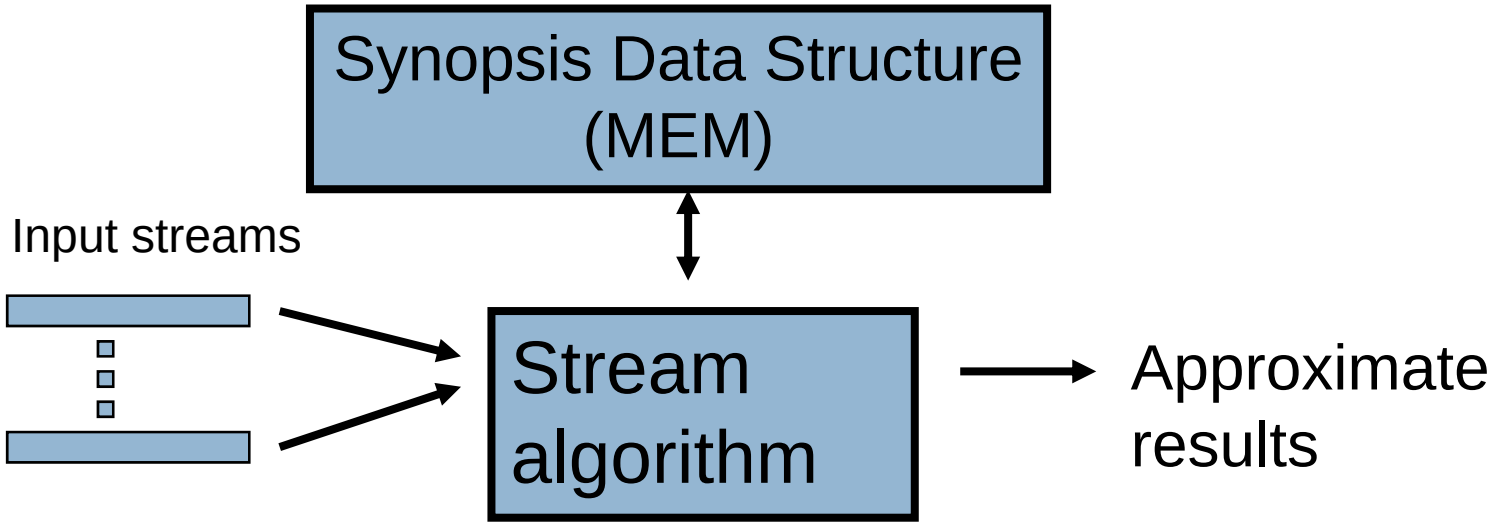
MEM: poly(1/ε, log N)

Query/Update: poly(1/ε, log N)

N: stream size or window size

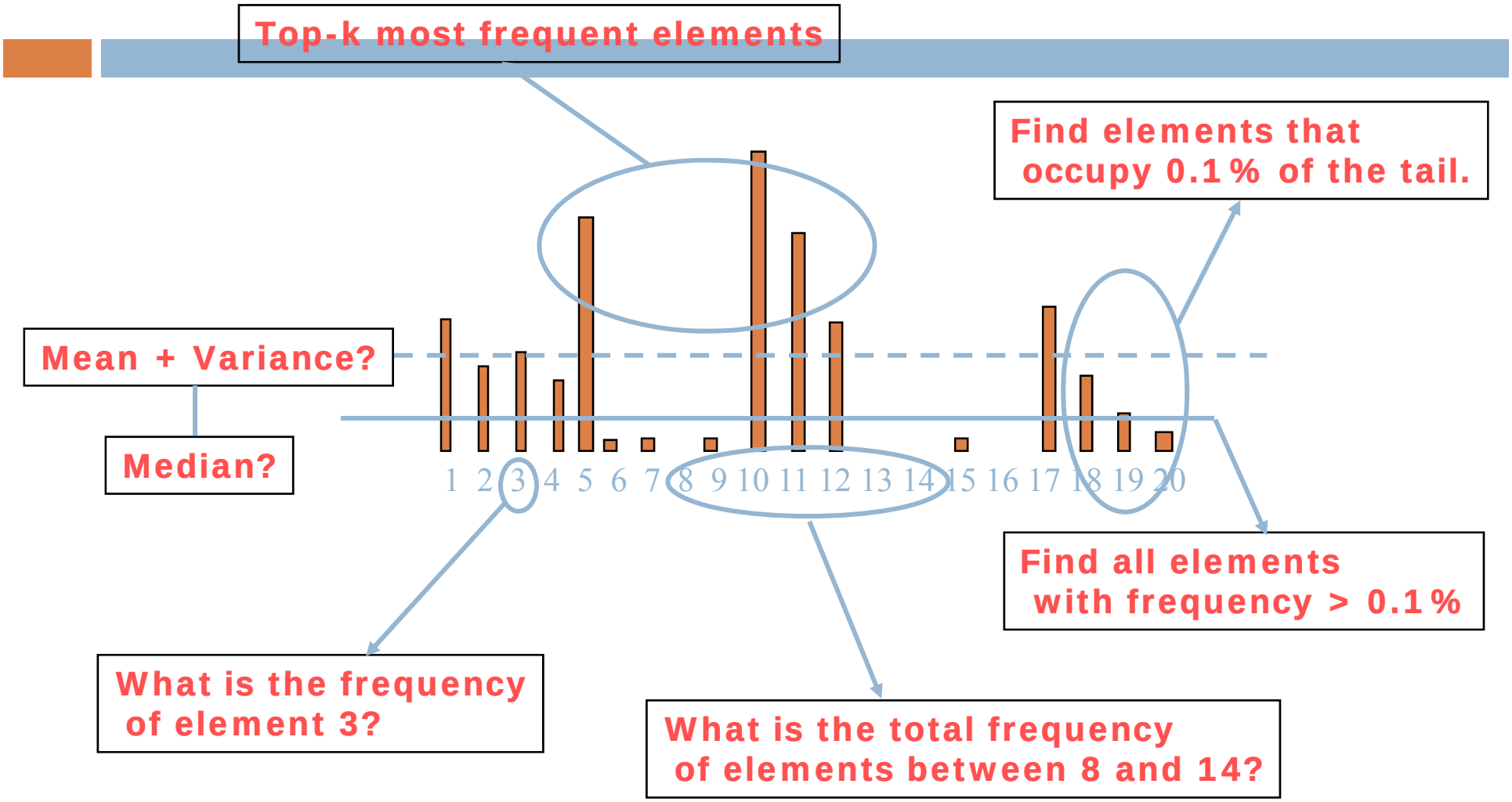
ε: error parameter

Data Models

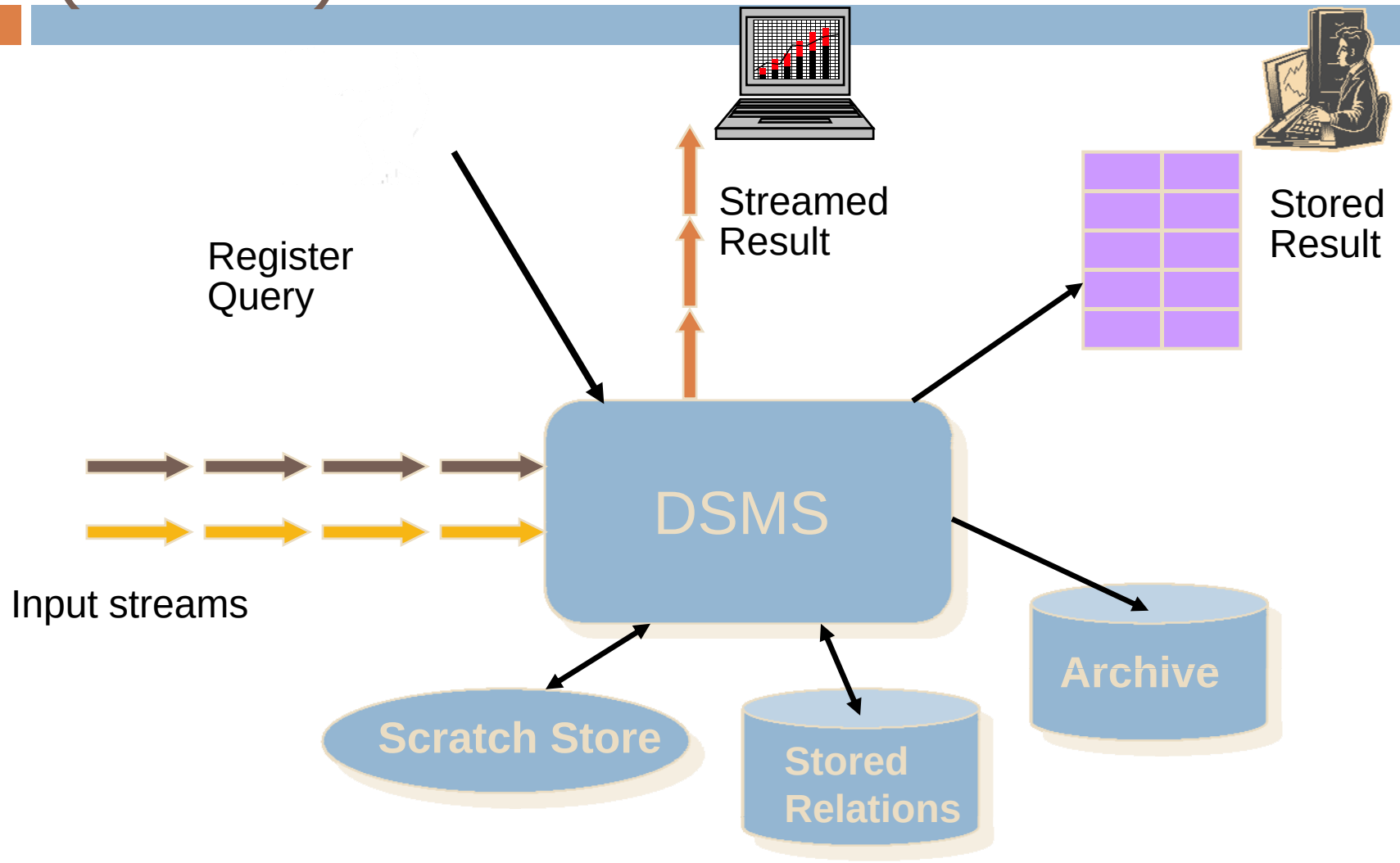


- According to Data characteristics
  - Time series model
  - Cash-register model
  - Turnstile model
- According to query range
  - Landmark model
  - Sliding-window model
  - Damped-window model

Frequency Related Problems



Data Stream Management System (DSMS)



DBMS vs. DSMS

- Database Systems
  - Model: persistent relations
  - Relation: tuple set/bag
  - Data update: modifications
  - Query: transient
  - Query Answer: exact
  - Query Evaluation: arbitrary
  - Query Plan: fixed
- Data Stream Systems
  - Model: transient relations
  - Relation: tuple sequence
  - Data update: appends
  - Query: persistent
  - Query Answer: approximate
  - Query Evaluation: one pass
  - Query Plan: adaptive

Some Systems Developed at Labs

- Aurora
  - Brow University, MIT, Brandeis University
  - http://www.cs.brown.edu/research/aurora/
- Niagara
  - University of Wisconsin
  - http://www.cs.wisc.edu/niagara/
- STREAM
  - Stanford University
  - http://infolab.stanford.edu/stream/
- TelegraphCQ
  - U. C. Berkley
  - http://telegraph.cs.berkeley.edu/telegraphcq/v0.2/

From Laboratories to Industries

- Aurora → StreamBase
  - http://www.streambase.com/
- STREAM → Coral8
  - http://www.coral8.com/
- TelegraphCQ → Truviso
  - http://www.truviso.com/
- CEP: Complex Event Processing
- RT-BI: Real Time Business Intelligence

Our Research Interests

	2003	2004	2005	2006	2007	2008
Survey		JOS				
Frequent pattern	CIKM	VLDB	DASFAA, WAIM	IS		
Clustering, Density estimate	DASFAA			SDM,WAIM,JCST	KAIS,ICDE,JOS	
Stream system		JCRD	APWEB	DASFAA	FCSC	ICDE
Aggregate query			DASFAA	ICDE, JOS	ICDE	
Complex Data			ICDE			ICDE
Uncertainty						VLDB

Frequent Pattern Mining

	2003	2004	2005	2006	2007	2008
Survey		JOS				
Frequent pattern	CIKM	VLDB	DASFAA, WAIM	IS		
Clustering, Density estimate	DASFAA			SDM,WAIM,JCST	KAIS,ICDE,JOS	
Stream system		JCRD	APWEB	DASFAA	FCSC	ICDE
Aggregate query			DASFAA	ICDE, JOS	ICDE	
Complex Data			ICDE			ICDE
Uncertainty						VLDB

Synopsis Maintenance & Counting– Background

- Data items: $<id,+/->$
  Queries: $count(id)$
  - Usually, **hot-spots**, i.e., the items with highest $count(id)$'s, are most interested.
- Existing approaches
  - Sketch-based [CCFC02]
    - Space complexity: $\Omega(k/\varepsilon^2 \log n)$
  - Group-test [CM03]
    - Space complexity: $O(k(\log k+\log 1/\delta)\log M)$

hCount: Structure and algorithm

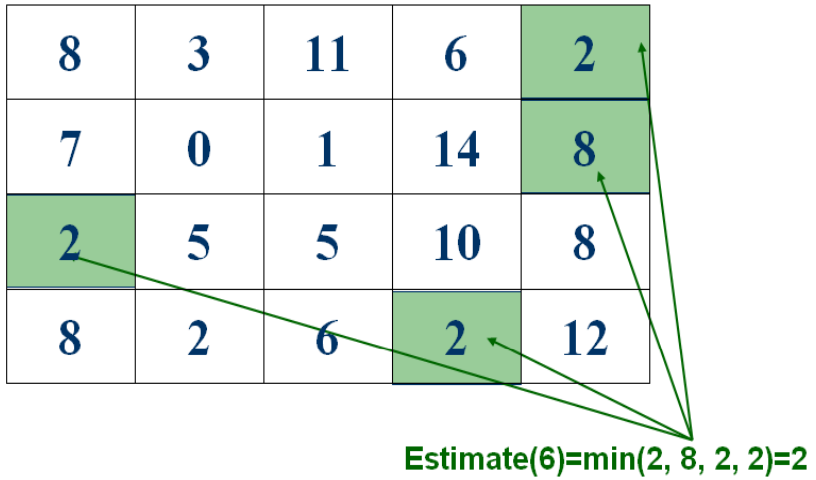
[Jin, CIKM03]

- Synopsis data structure
  - Hash table: $S[m][h]$, each item is a count（Bloom-filter + counters）
  - h hash functions: $H_1, H_2, \ldots, H_h$ (H_i: $[1{:}M] \rightarrow [1{:}m]$)
  - Maintenance
    - When an item arrives, the hash table is updated
    - When a count estimation query arrives
      - Select one candidate item in each line
      - Return the candidate item with lowest value

hCount: An example

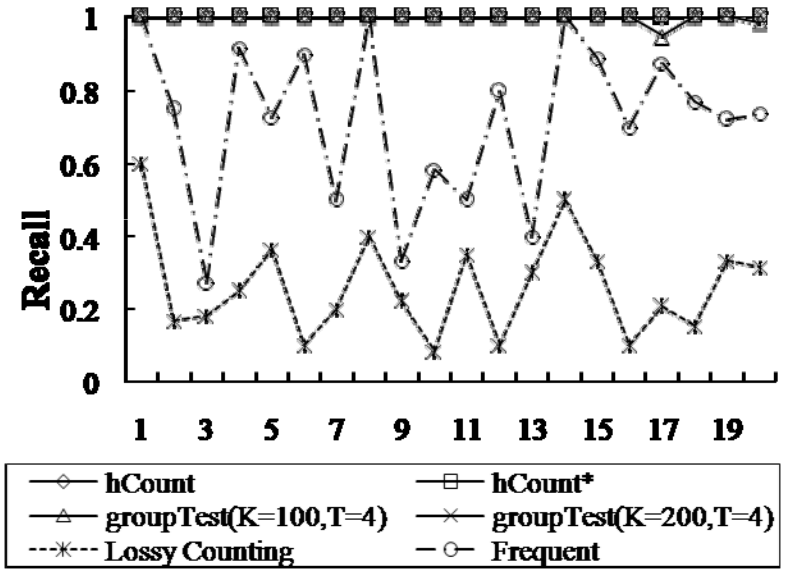
Steps: (1−6):

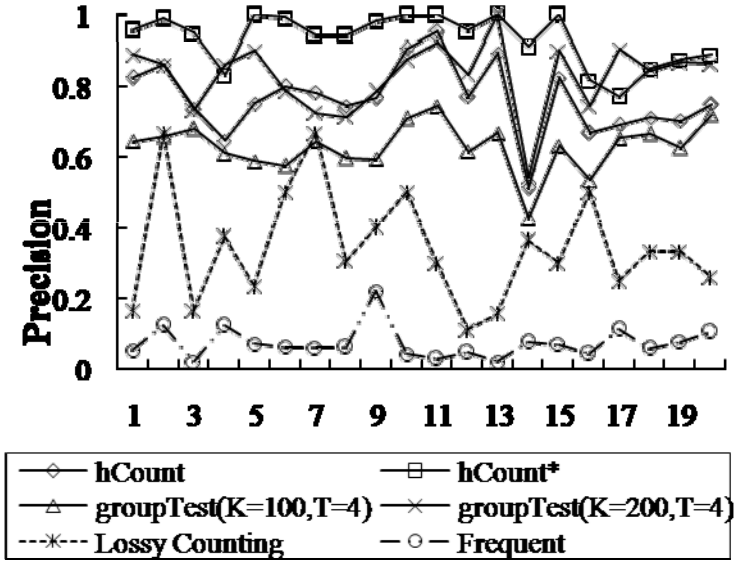
1. Hash table at t_4
2. At t_5 : insert(9)
3. At t_6 : delete(6)
4. Hash table at t_{38}
5. Estimating item 9
6. Estimating item 6



hCount: Conclusion

- Low space complexity: $O(\varepsilon^{-1} \log M)$
- Fast per-tuple processing cost
- High recall and precision





Frequent Pattern (FP) Mining:
Problem Description

- Frequent pattern mining is a basic problem in data mining
  - Given a set of transactions, each of which is a set of items
    - The **support** of an itemset is the frequency of the transactions contain the itemset
  - To find out all the itemsets whose support is greater than the threshold
- Frequent pattern mining over data streams is different to FP mining on static databases
  - The data stream can only be scanned once
  - For combinational explosion, previous FP-mining methods cannot be extended to be applied on data streams
- Basic idea
  - Frequent item mining is the first step of FP mining
  - Existing methods can be classified as:
    - False positive methods, and
    - False negative methods

FP Mining – False Positive *vs.* False Negative

- False positive methods
  - The itemsets whose support is lower than the threshold may be returned
  - ✘ **Too many false positive results may greatly affect the reliability of the mining process**
    - Previous methods (based on error control parameter ε) all report a large amount of false positive results
    - A large amount of combinations of items should be considered due to the false positive items/itemsets
    - May result in huge consumptions of memory storage
- False negative methods
  - May ignore itemsets whose support is above the threshold
  - ✓ **Higher accuracy**
  - ✓ **Much fewer storage consumption**

FDPM (Frequent Data-stream Pattern Mining)

[Yu, VLDB04]

- Mining frequent patterns (items/itemsets) from a high speed transactional data stream.
  - A large data item domain (I)
  - A virtually unbounded sequence of a data stream (N)
  - Limit memory space (M)
  - One-scan of data
- The main difficulty of frequent itemsets mining
  - The possible number of itemsets is 2^I-1
  - When N becomes very large, the possibility of an itemset to be frequent becomes larger, and it is difficult to track with limited memory.
- The technique of frequent itemsets mining is largely dependent on the technique developed for frequent items mining.
  - Approximate mining with two parameters ε and δ, using a synopsis.

FDPM: Basic Ideas

- Mining frequent items
  - Using dynamic error parameter ε_n (decided by minimum support and probability parameter δ)
  - δ is used to control the memory consumption
    - Space complexity is related to $\ln(1/\delta)$, not $1/\varepsilon$
  - Pruning technique: Chernoff bound
- Extension: Mining frequent itemsets
  - Using rules to reserve frequent itemsets

FDPM – Summary

- FDPM: One-scan false-negative algorithm for frequent pattern mining
  - The frequent itemsets that are falsely dropped are probabilistically bounded
    - The accuracy of frequent item mining is theoretically guaranteed
    - The empirical bound of accuracy for frequent itemset mining is investigated
  - Empirical result show the effectiveness and efficiency of our FDPM method

Clustering and Density estimation

	2003	2004	2005	2006	2007	2008
Survey		JOS				
Frequent pattern	CIKM	VLDB	DASFAA, WAIM	IS		
Clustering, Density estimate	DASFAA			SDM,WAIM,JCST	KAIS,ICDE,JOS	
Stream system		JCRD	APWEB	DASFAA	FCSC	ICDE
Aggregate query			DASFAA	ICDE, JOS	ICDE	
Complex Data			ICDE			ICDE
Uncertainty						VLDB

Clustering using GPU

[Cao, WAIM06]

- Graphics Processing Units
  - Parallel processing
  - SIMD
- Observation
  - K-means is a widely used clustering method
  - GPU can optimize two time-cost operations
    - Distance calculation
    - Distance comparison
  - The above operators can be optimized by GPU

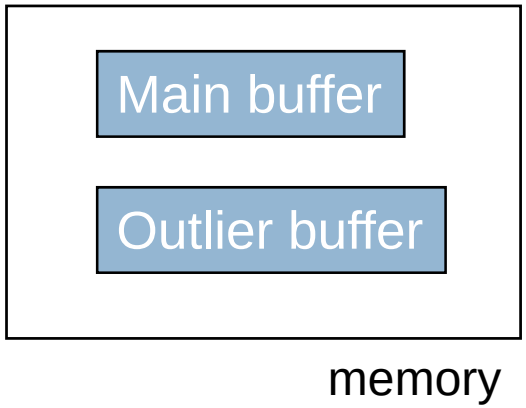
Clustering an Evolving Stream

[Cao, SDM06]

- The clusters will change their shapes with time going on.
  - Important to report the changes.
- Our method: Density-based clustering method
  - Capable of summarizing clusters with arbitrary shape.
- Contributions
  - Can handle arbitrary shape, not only spheres.
  - Capable of reporting outliers
  - Using *outlier-buffer to hasten the processing*.
  - achieves consistently high clustering

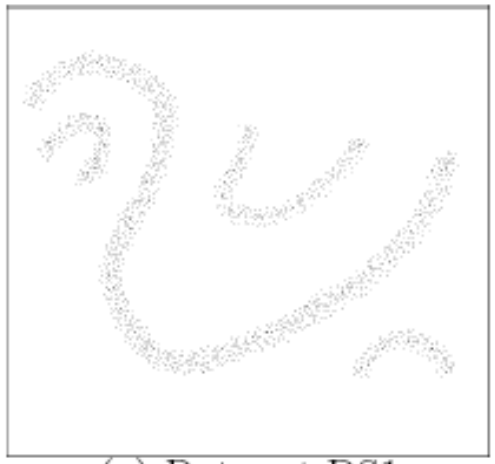
Clustering an Evolving Stream: Algorithm

- Structure
  - C-Micro-cluster:
    - used for potential clusters
    - In the main buffer
  - O-micro-cluster:
    - potential outliers
    - In the outlier buffer

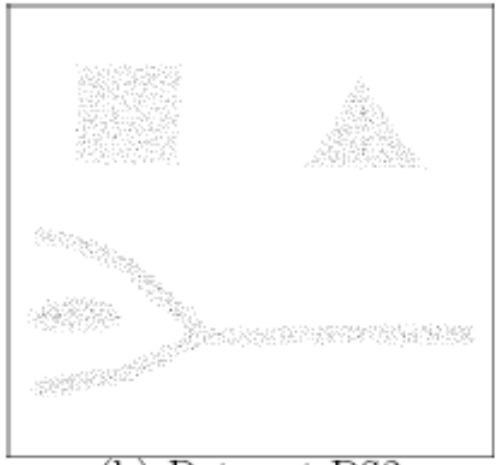


- Merge operator
  - For each new pointer *p*
    - Try to find an existed micro-cluster in main buffer to merge
    - If failed, check whether it can be absorbed by an o-micro-cluster.
    - If step 2 successes, try to convert an o-micro-cluster to a c-micro-cluster
    - If Step 2 failed, this point is created as an o-micro-cluster.

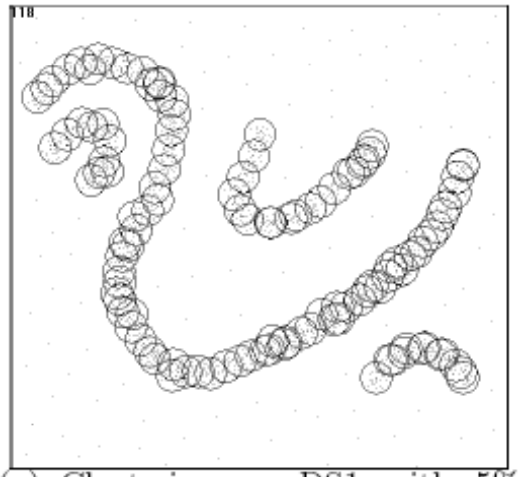
Clustering an Evolving Stream: Experiments



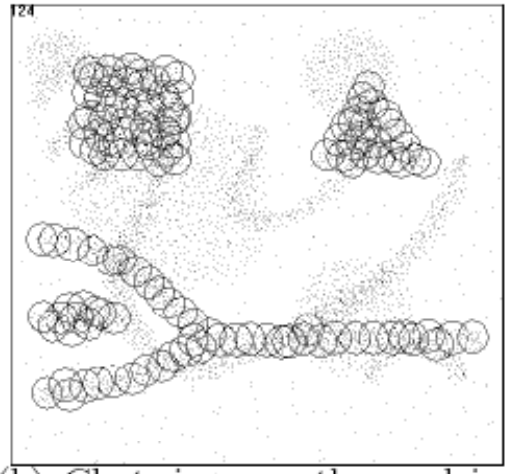
(a) Data set DS1



(b) Data set DS2



(a) Clustering on DS1 with 5% noise



(b) Clustering on the evolving stream EDS with 5% noise(t−20)

Clustering over Sliding Windows–Problem Description

- For continuously arriving multi-dimensional (d-dim'l) tuples $\mathbf{X}_i(x_i^1, x_i^2, \ldots x_i^d)$, analyzing
  - The clusters on the last N tuples
  - The emerge, development and trends of these clusters
  - Previous methods:
    - Cannot efficiently process over sliding windows, or
    - Cannot accurately analyze the evolving of a specific cluster

SWClustering: Clustering over Sliding Windows

[Zhou, KAIS07]

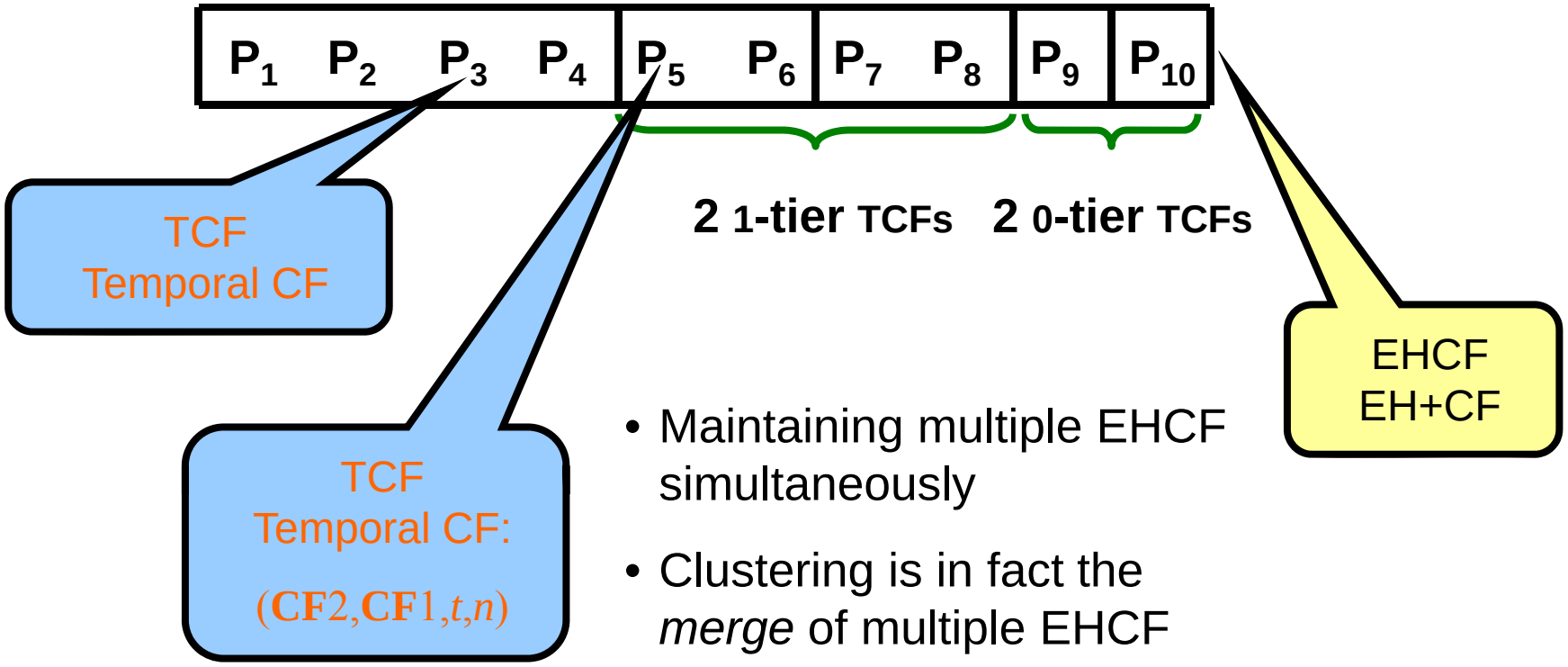
- Goal:
  - Clustering over sliding window
  - Tracking the evolving of clusters

SWClustering: Clustering over Sliding Windows

[Zhou, KAIS07]

- Basic idea
  - Integrate information of time in cluster features
    - Maintain sufficient information about the evolving of a cluster
  - Embed cluster features (CF) in exponential histograms (EH)
    - $\{EH_i(\{CF_j^i\})\}$
    - Can maintain the sufficient information for clustering with approximation factor $1+\varepsilon$
  - Compared with pyramidal technology [AHWY03] $SS_i(\{CF_j\})$:
    - ✓ Need not to store snapshot SS
    - ✓ Since a specific cluster can be retrieved to be analyzed alone, it is more flexible

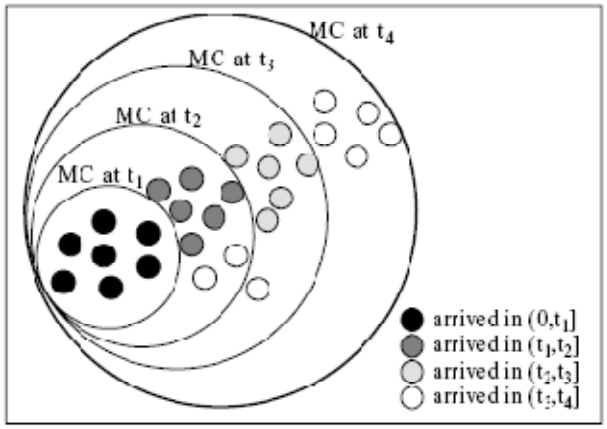
SWClustering: Data Structure



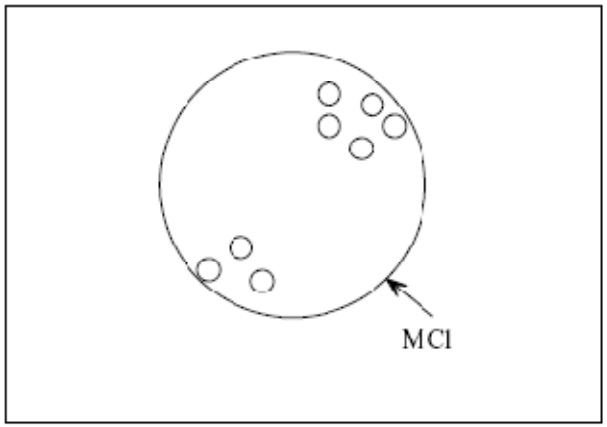
- Maintaining multiple EHCF simultaneously
- Clustering is in fact the *merge* of multiple EHCF
- Efficiency can be further improved by later merging of EHCF

SWClustering: Experiments

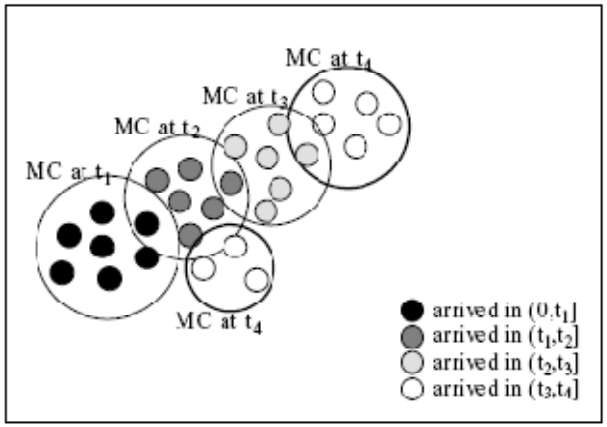
- SWClustering v.s. CluStream
- Economic space consumption



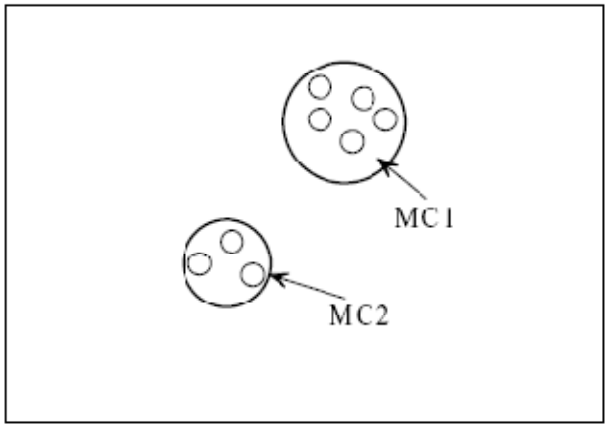
(a) The formation of micro clusters in CluStream



(b) The micro clusters formed by recent records in CluStream



(c) The formation of microclusters in our sliding-window-based method



(d) The micro clusters formed by recent records in our sliding-window-based method

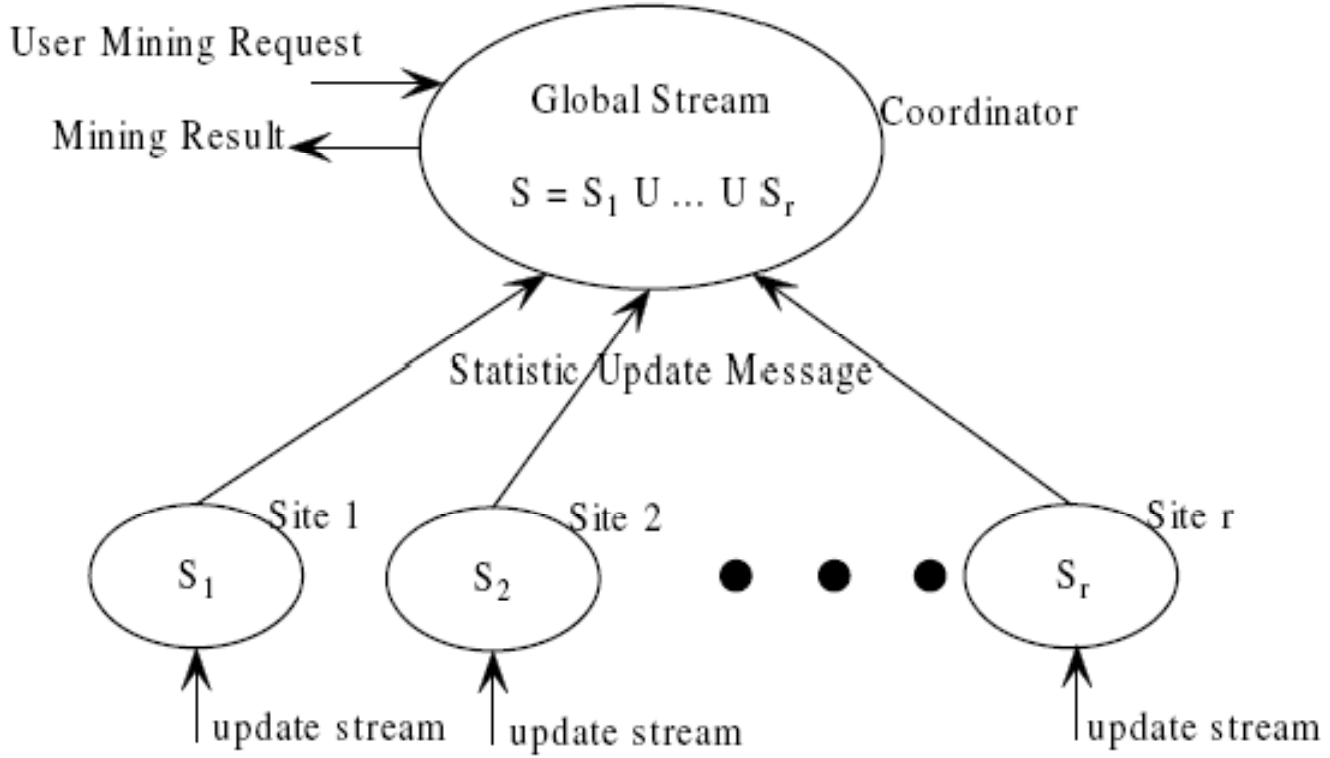
Clustering Analysis – Summary

- SWClustering is more accurate and efficient than snapshot-based methods
- Due to the embedding of CF in EH, the analysis is more flexible
- Theoretical analysis show that the space consumption is limited
- Empirical result show that this method is efficient for clustering a data stream

Distributed Clustering: Scenario

[Zhou, ICDE07]

- Goal: Reducing the communication cost



Expectation-Maximization (EM) Algorithm

1. Initialization: Initialize the mixture model parameters $(w_j^0, u_j^0, \Sigma_j^0)$, $j = 1, ..., k$, and iteration step $i = 0$.

2. Repeat

   (a) E-Step. For each record x, compute the membership probability of x in each cluster: $Pr(j|x) = \frac{w_j^i \cdot Pr(x|j)}{Pr(x)}$, $j = 1, ...k$.

   (b) M-step. Update the parameters of mixture model: $w_j^{i+1} = \frac{1}{|D|} \sum_{x \in D} Pr(j|x)$,
   $$u_j^{i+1} = \frac{\sum_{x \in D} x \cdot Pr(j|x)}{\sum_{x \in D} Pr(j|x)},$$
   $$\Sigma_j^{i+1} = \frac{\sum_{x \in D} Pr(j|x)(x - u_j^{i+1})(x - u_j^{i+1})^T}{\sum_{x \in D} Pr(j|x)},$$
   $j = 1, ..., k$.

   (c) Compute the log likelihood:
   $$E^{i+1} = \sum_{x \in D} \log Pr(x)$$
   $$= \sum_{x \in D} \log \sum_{j=1}^{k} w_j^i \cdot Pr(x|j).$$

   Until $|E^i - E^{i+1}| \leq \varpi$, a user-defined parameter.

Distributed Clustering: Basic Steps

- Remote site processing
  - Divide & Conquer
  - Cluster the incoming data records only after they don't pass the test
- Coordinator Processing
  - Avoid local maximal --Over component population
  - In general, when many data records lead to almost equal posterior probability for any two components, it can be thought that these two components might be merged.

$$M_{merge}(i,j) = \frac{1}{(u_i - u_j)^T(\Sigma_i^{-1} + \Sigma_j^{-1})(u_i - u_j)}$$

Distributed Clustering: Summary

- A framework to cluster the distributed data streams with soft membership in consideration
- Event driven clustering for capturing the distribution of individual stream at remote site
- Can also be applied to a lot of related problems

Our Interests: Streaming system

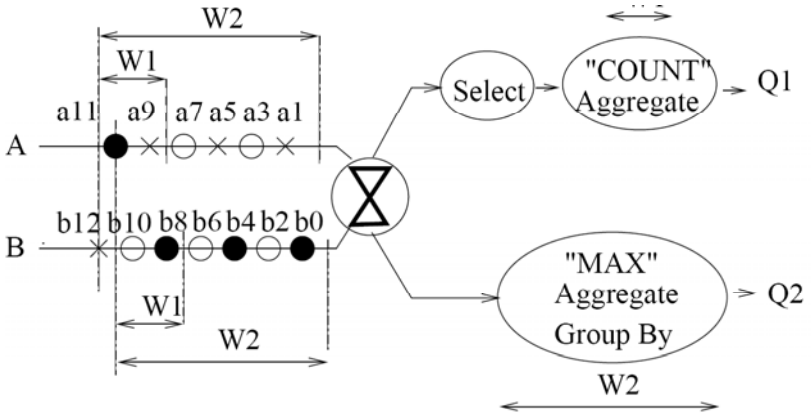
	2003	2004	2005	2006	2007	2008
Survey		JOS				
Frequent pattern	CIKM	VLDB	DASFAA, WAIM	IS		
Clustering, Density estimate	DASFAA			SDM,WAIM,JCST	KAIS,ICDE,JOS	
Stream system		JCRD	APWEB	DASFAA	FCSC	ICDE
Aggregate query			DASFAA	ICDE, JOS	ICDE	
Complex Data			ICDE			ICDE
Uncertainty						VLDB

Shared Window Joins: Definition

- Example

Q1: SELECT count(A.Id)
FROM A, B
WHERE A.Id=B.Id and A
WINDOW 1 min;

Q2: SELECT A. Id, MAX(B.value)
FROM A, B
WHERE A.Id=B.Id GROUP BY A. Id
WINDOW 1 hour;

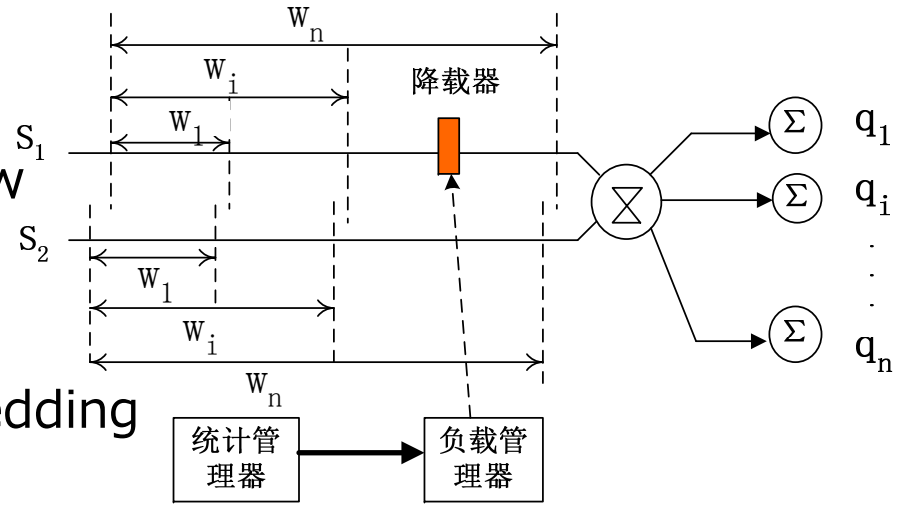


- Previous methods
  - MQT[HFAE03]: schedule tuples according to the window sizes
    - ✘ Cannot adapt to the bursty stream rates

Shared Window Joins: Load Shedding

[Yan, JCRD04]

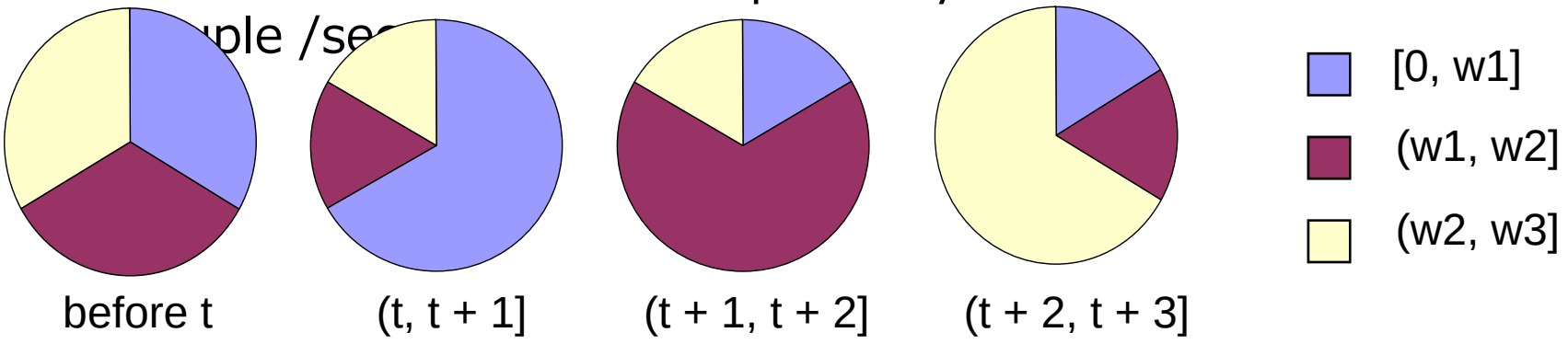
- Basic Idea (Load shedding)
  - When the query cannot be handled with current resources, parts of data is dropped to shed system's load
  - ✓ Can enhance system's throughput
  - ✘ Can only provide approximate results
- Steps
  - Deploy load shedding operators at every window
  - Two strategies
    - Uniform shedding
    - Small-window exact shedding



Shared Window Joins: Adaptive Scheduling

[Jin, FCSC07]

- Basic Idea
  - Schedule tuples according to the count of data items, not the time-base window sizes
  - Can reduce the response time
- Example
  - Consider three queries Q1, Q2, Q3 over two streams A, B, whose window sizes w1, w2, and w3 are 1 second, 2 second and 3 seconds respectively. The rate of stream A is ple /se



Shared Window Joins: Adaptive Scheduling

- The method
  1. All uncompleted tuples are reserved in a BUFFER.
  2. Monitoring the number of tuples in the BUFFER.
     1. Normally, the number of tuples is in low level. When it exceeds a threshold, a burst rises.
        1. Generate a real-time scheduling plan according to tuple distribution.
        2. Schedule future tuples and buffered tuples by the new scheduling plan.
- Analysis
  - The real-time scheduling plan is optimal for bursty tuples.
  - Cost to generate a scheduling plan can be ignored.
  - It much outperforms MQT (best previous method) method for applications where rate changes violently
  - It owns comparable performance with MQT method when rate is steady

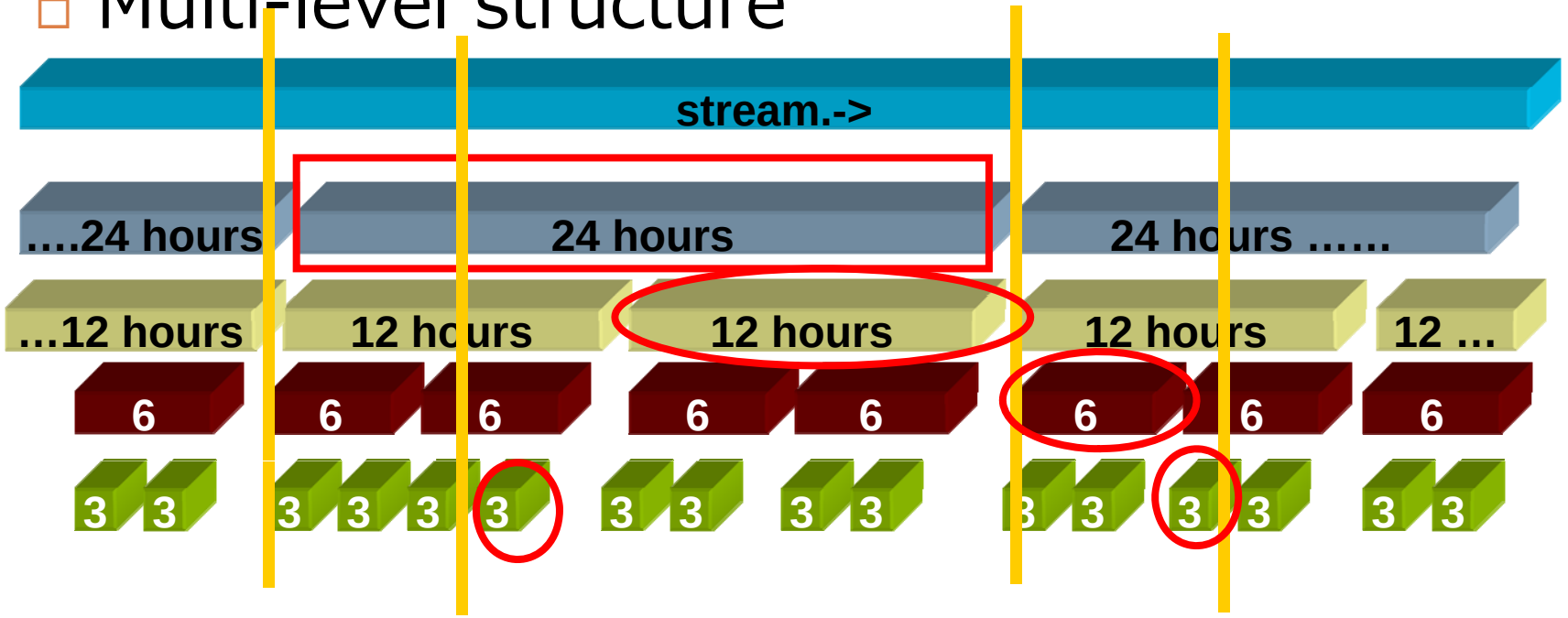
SMART: Background

[Zhou, ICDE08]

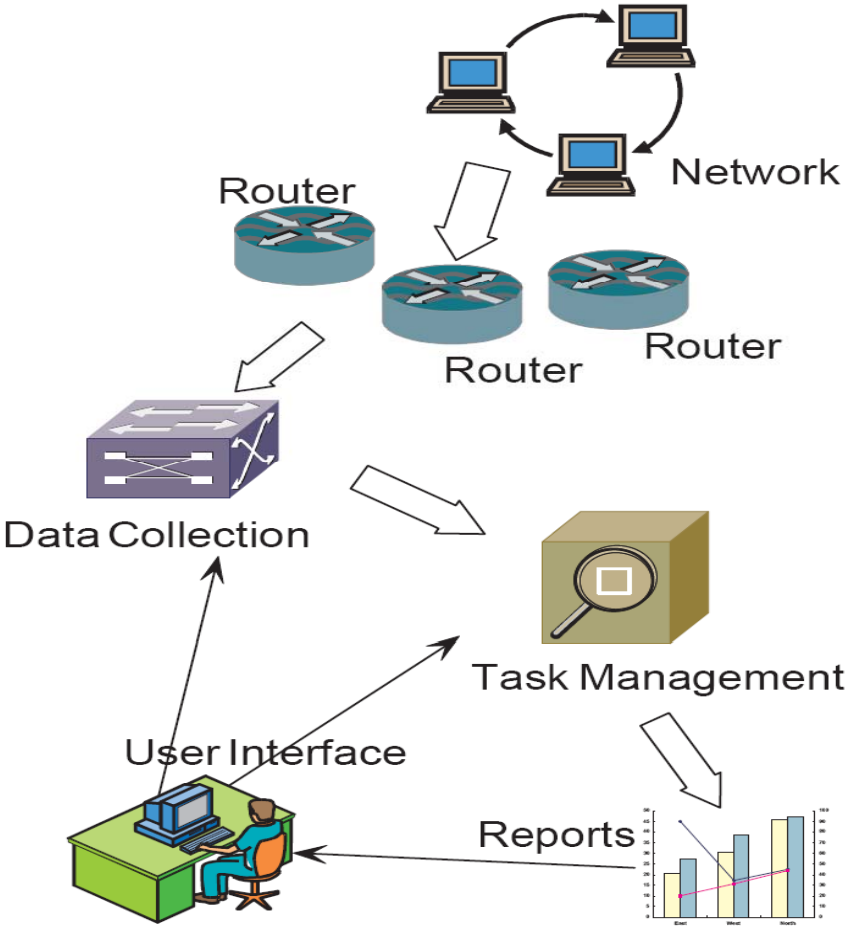
- SMART: A System for Online Monitoring Large Volumes of Network Traffic, a project from Shanghai Telecomm
- More and more business of Telecomm are run on IP network
- Acquire network traffic patterns and take quick actions on abnormalities
- Problems of current systems in telecom companies
  - Based on Flow-tools in a disk-based offline manner
  - Inefficiency, too long response time
- Solutions — streaming algorithms
  - Memory based
  - Approximate with error guarantee
  - Abstract in formations into sketches

SMART: Sliding Window Frequent Top-k

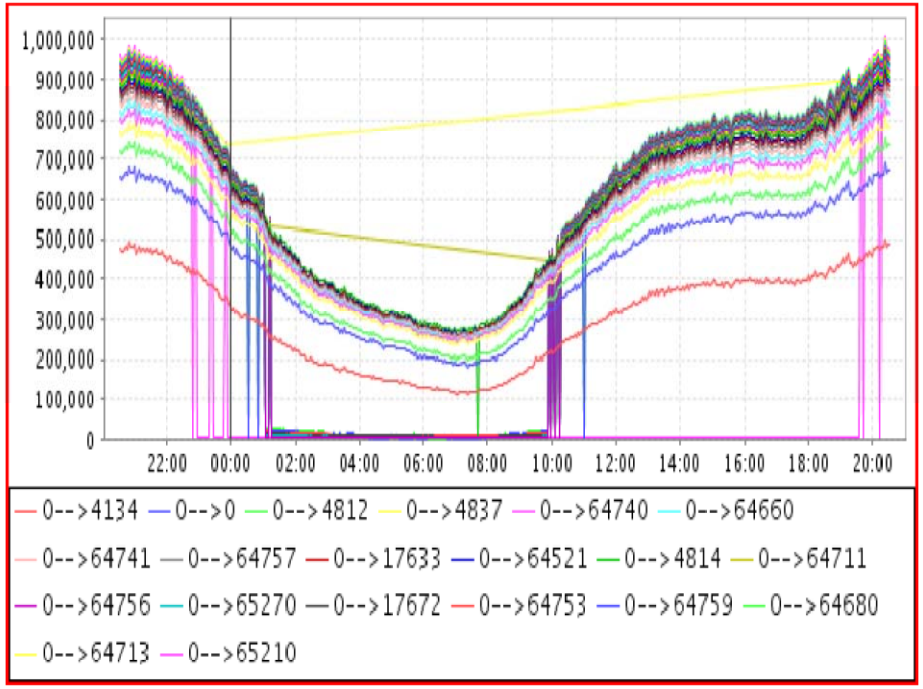
- Multi-level structure

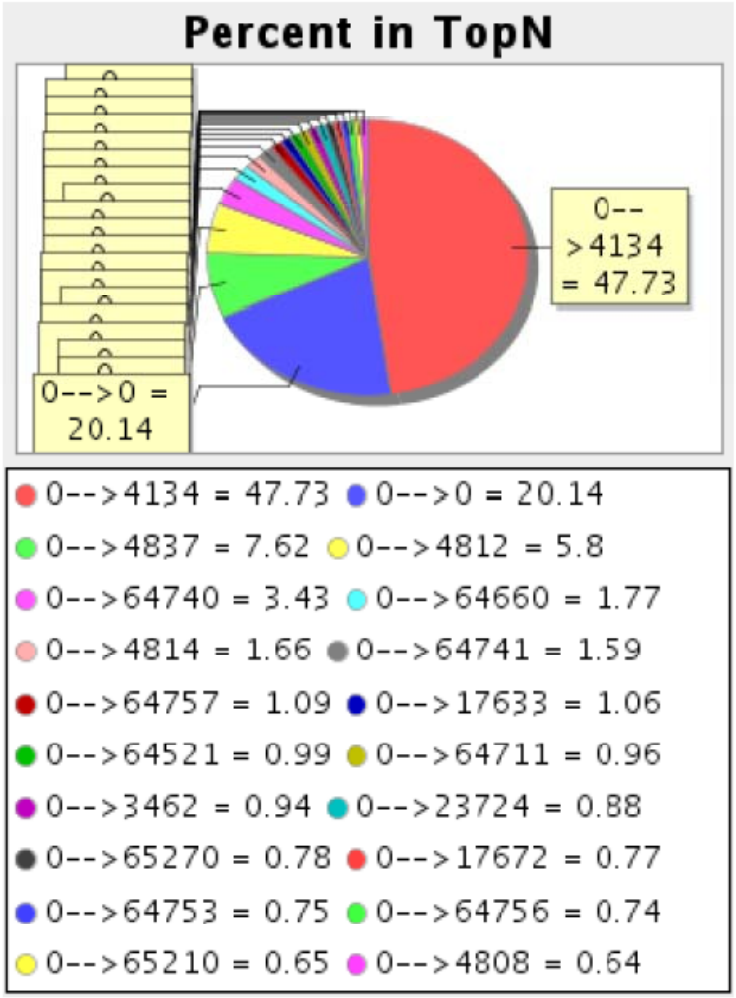


SMART: System Architecture



SMART: Visualizations





Our Interests: Streaming System

	2003	2004	2005	2006	2007	2008
Survey		JOS				
Frequent pattern	CIKM	VLDB	DASFAA, WAIM	IS		
Clustering, Density estimate	DASFAA			SDM,WAIM,JCST	KAIS,ICDE,JOS	
Stream system		JCRD	APWEB	DASFAA	FCSC	ICDE
Aggregate query			DASFAA	ICDE, JOS	ICDE	
Complex Data			ICDE			ICDE
Uncertainty						VLDB

Aggregate Monitoring: Problem

- Aggregate functions: sum, avg, min, max, …
- Applications of online monitoring data streams:
  - Networks traffic management, trend-related analysis, web-click streams analysis, stocks exchange, e-mail and news articles, and sensor networks.
- However, existing methods suffer several problems:
  - Expensive cost of time O(m)
  - Aggregate functions are restricted to be first-order statistic or monotonic with respect to the window size.

Aggregate Monitoring: Problem

- Assuming that F is an aggregate function, w_i is a sliding window with given length, and T(w_i) is the threshold of F(w_i) given by users beforehand. The result is reported on w_i when stream X is updated and if F(w_i) ≥T(w_i).

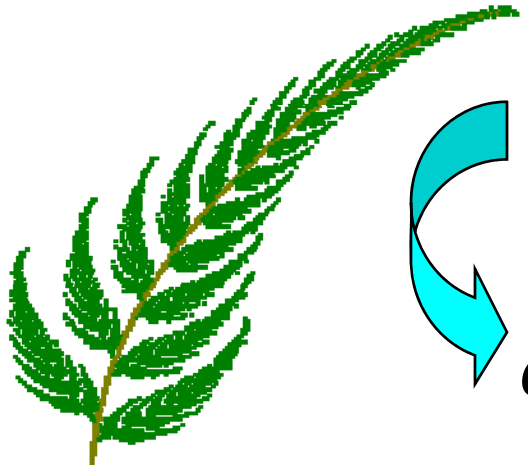
Fractal-based Approach: contributions

[Qin, ICDE06]

- Incorporate the tool of fractal analysis and can process data which does not obey the exact fractal models.
- The monotony property of aggregate monitoring is revealed and monotonic search space is built to decrease the time overhead from O(m) to O(log m).
- Propose a novel synopsis, called Inverted Histogram (IH), and provide the error bound for IH

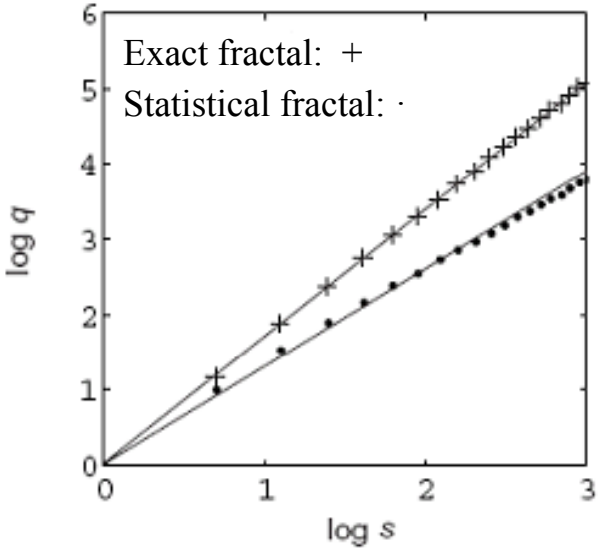
Fractal-based Approach: Fractal

- Power Law Scaling Relationship
  - When a quantitative property, *q*, is measured on scale *s*, its value depends on *s* according to the following power law scaling relationship:



$$d = \frac{\log q}{\log s}$$

$$d = \frac{\log F(w_i)}{\log w_i}$$

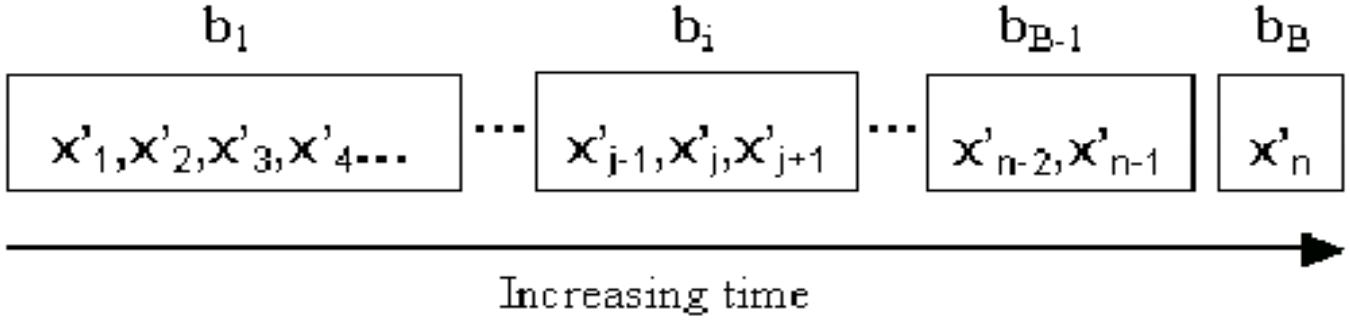


Fractal-based Approach: Inverted Histogram

- x_i' is suffix sum of stream $x_1 \cdots x_n$

$$x_i' = \sum_{j=i}^{n} x_j$$

- Relative Error is bounded by **d**



- Time cost: $O(n \log(nR)/\log(1+\delta))$
- Space cost: $O(\log(nR)/\log(1+\delta))$

Fractal-based Approach: Inverted Histogram

- Histogram building procedure
  - When x_n comes
  - Add a new bucket
  - For all buckets from the latest to the oldest one
    - Update each bucket
    - Merge two consecutive buckets;
  - End For

1: increase bucket number B by 1;
2: $j \leftarrow B$;
3: create a new bucket for x_n;

5: add x_n to both b_i^a & b_i^b;

6: If $b_i^b \leq (1+ d)b_j^a$ then
7: $b_j^b \leftarrow b_i^b$;
8: add width of b_i to width of b_j ;
9: delete b_i ;
10: decrease bucket number B by 1;

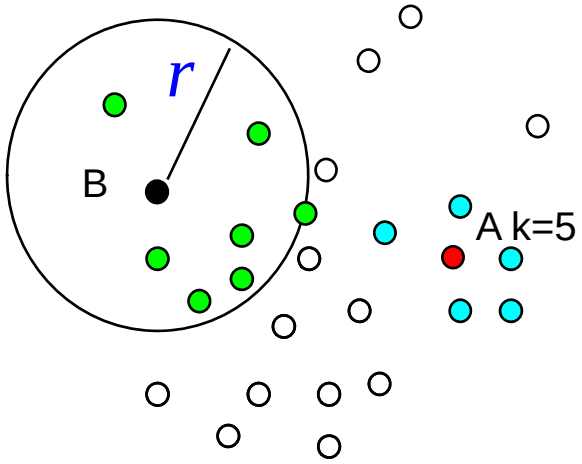
Fractal-based Approach:
Conclusions

- A novel method for monitoring aggregates of multi-granularity windows over a data stream
- Fractal analysis is introduced to transform the original data stream to an intermediate one
- Inverted histogram is adopted as the core synopsis for our monitoring tasks with guaranteed space and error

kNN monitoring

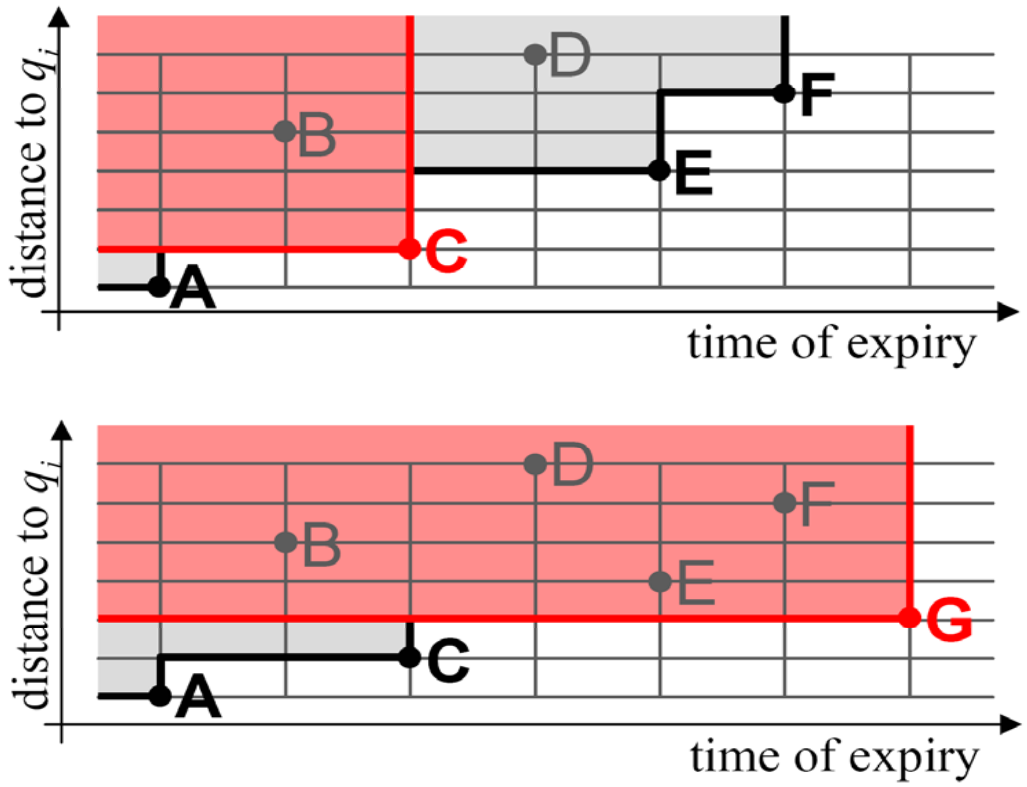
[Böhm, ICDE07]

- Two scenarios
  - Network Monitoring
    - Administrator specifies suspicious packages, the system is constantly surveyed for similar packages
  - Advertisement
    - Users specify property of interested products or news, the system informs users when new items fit their requirements
- Similarity query
  - Range query
  - K-NN query
- Basic idea
  - skyline



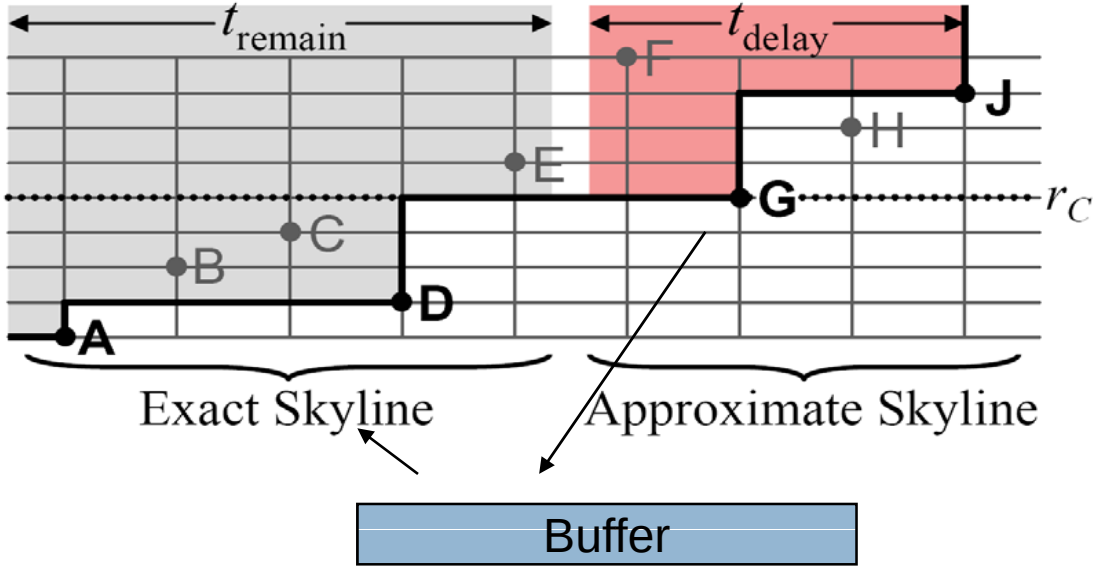
kNN monitoring: Object Maintenance

- Skyline

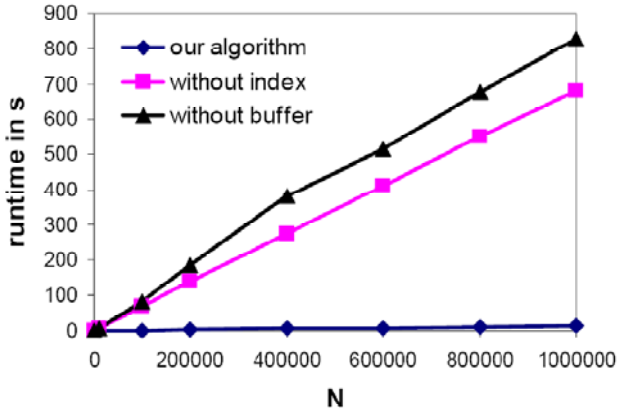


kNN monitoring: Object Delaying

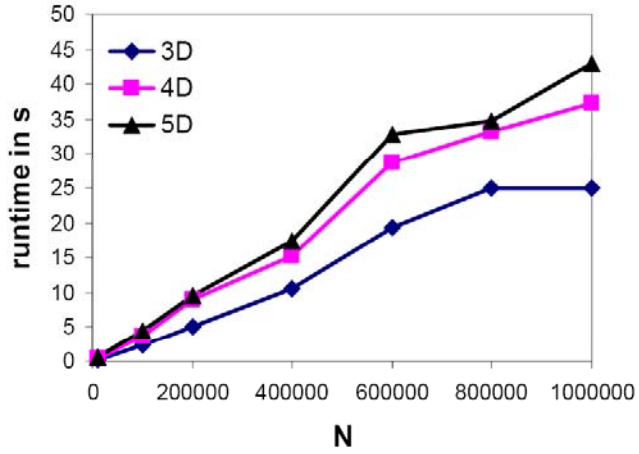
- Approximate and exact skyline
  - Buffer : size |B|, in life span order
  - Exact skyline: stores skyline points from approximate skyline
  - Approximate skyline: stores the address of skyline point



kNN monitoring: Experimental Results



(f) Effects of Query Indexing and Object Delaying.



(h) Scalability w.r.t d.

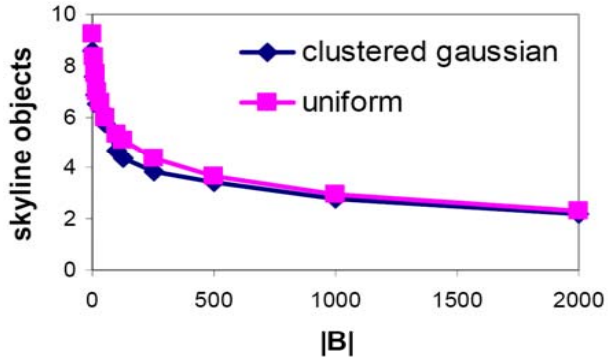


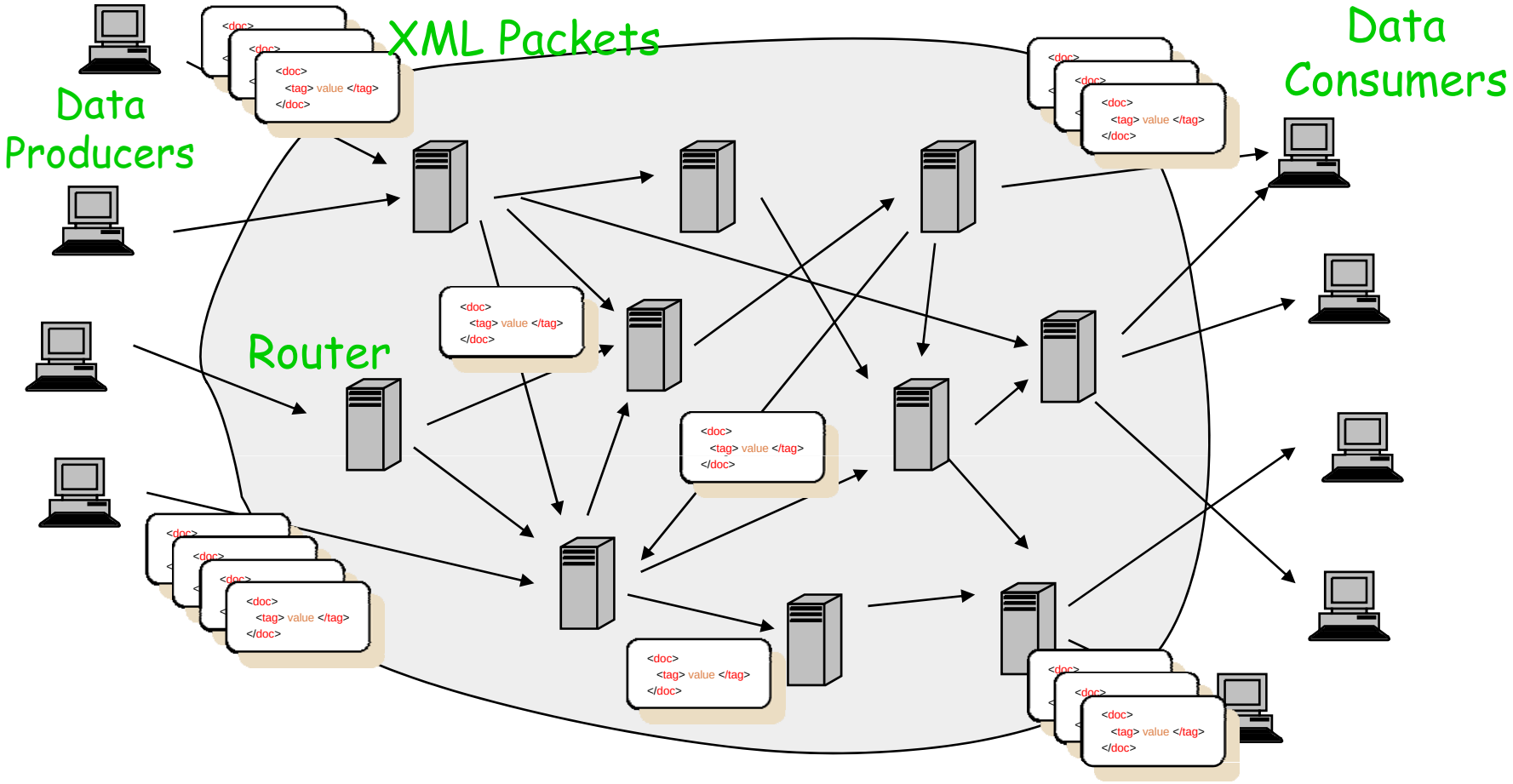
Figure 9: Buffer Size.

Our Interests: Complex data

	2003	2004	2005	2006	2007	2008
Survey		JOS				
Frequent pattern	CIKM	VLDB	DASFAA, WAIM	IS		
Clustering, Density estimate	DASFAA			SDM,WAIM,JCST	KAIS,ICDE,JOS	
Stream system		JCRD	APWEB	DASFAA	FCSC	ICDE
Aggregate query			DASFAA	ICDE, JOS	ICDE	
Complex Data			ICDE			ICDE
Uncertainty						VLDB

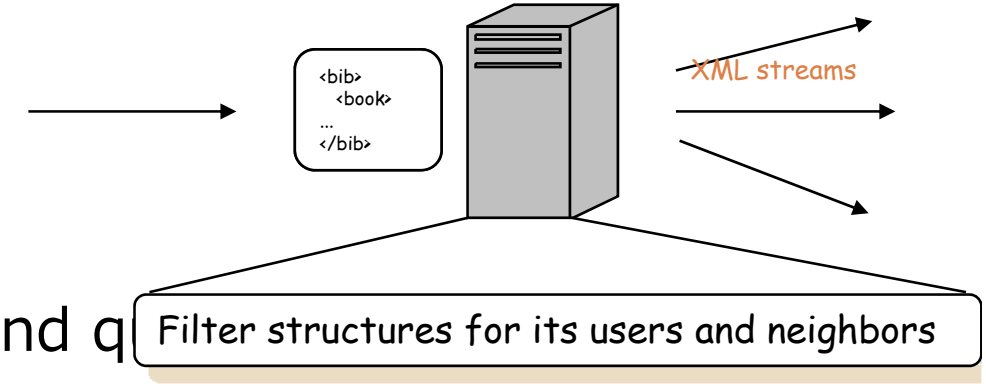
XML Streaming: Motivation

[Gong, ICDE05]



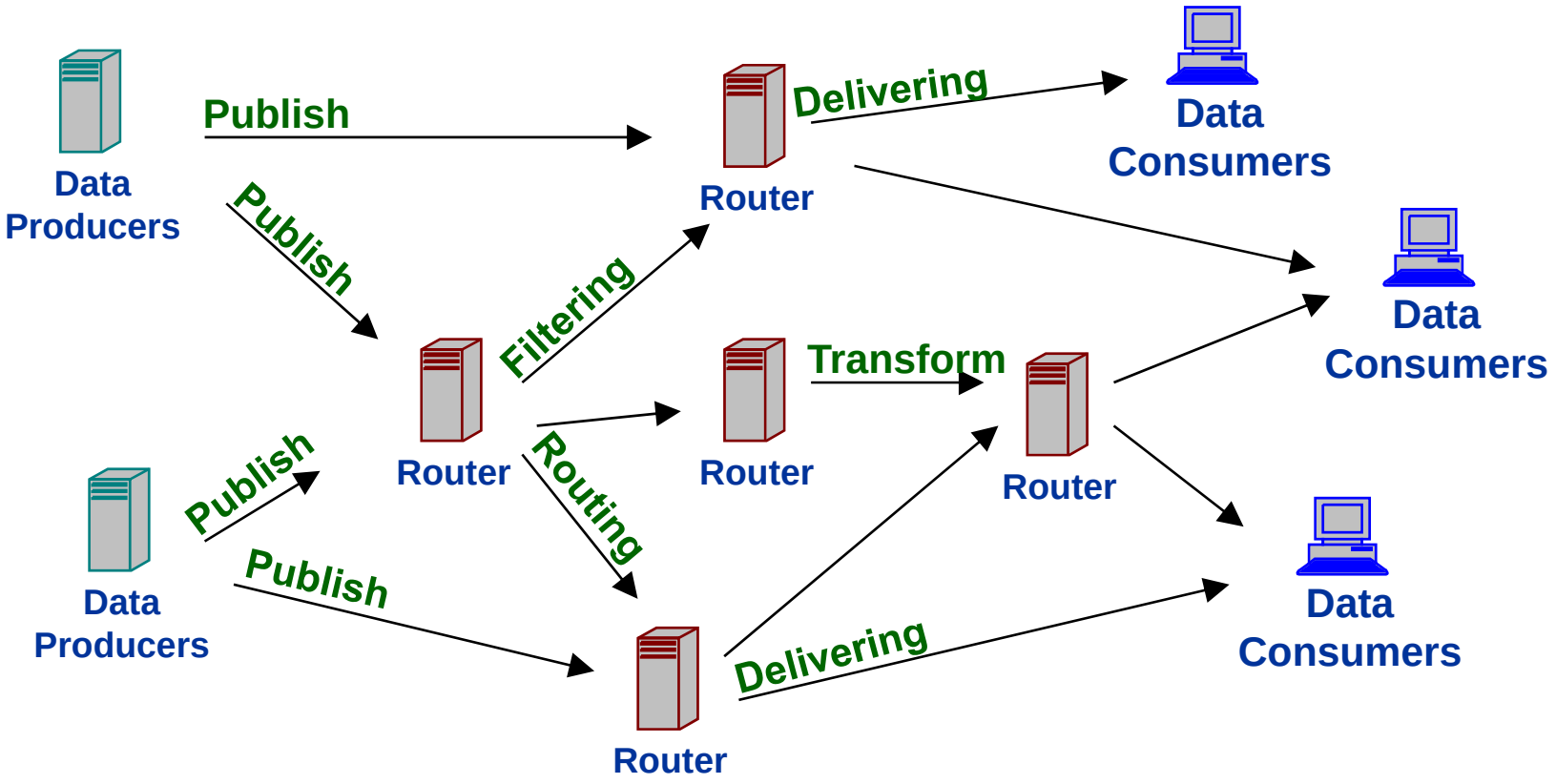
XML Streaming: Content Routing

- Problem
  - Each node is a router, which disseminates incoming XML packets to its neighbors or users.
  - Each router maintain a route structure for each user, which is used to determine if the XML packet will be matched by user's queries.



- Challenges
  - Large number of users and q
  - Changes of queries
- Basic Idea
  - Using bloom filter to build the route table

XML Stream Processing: Application Scenario

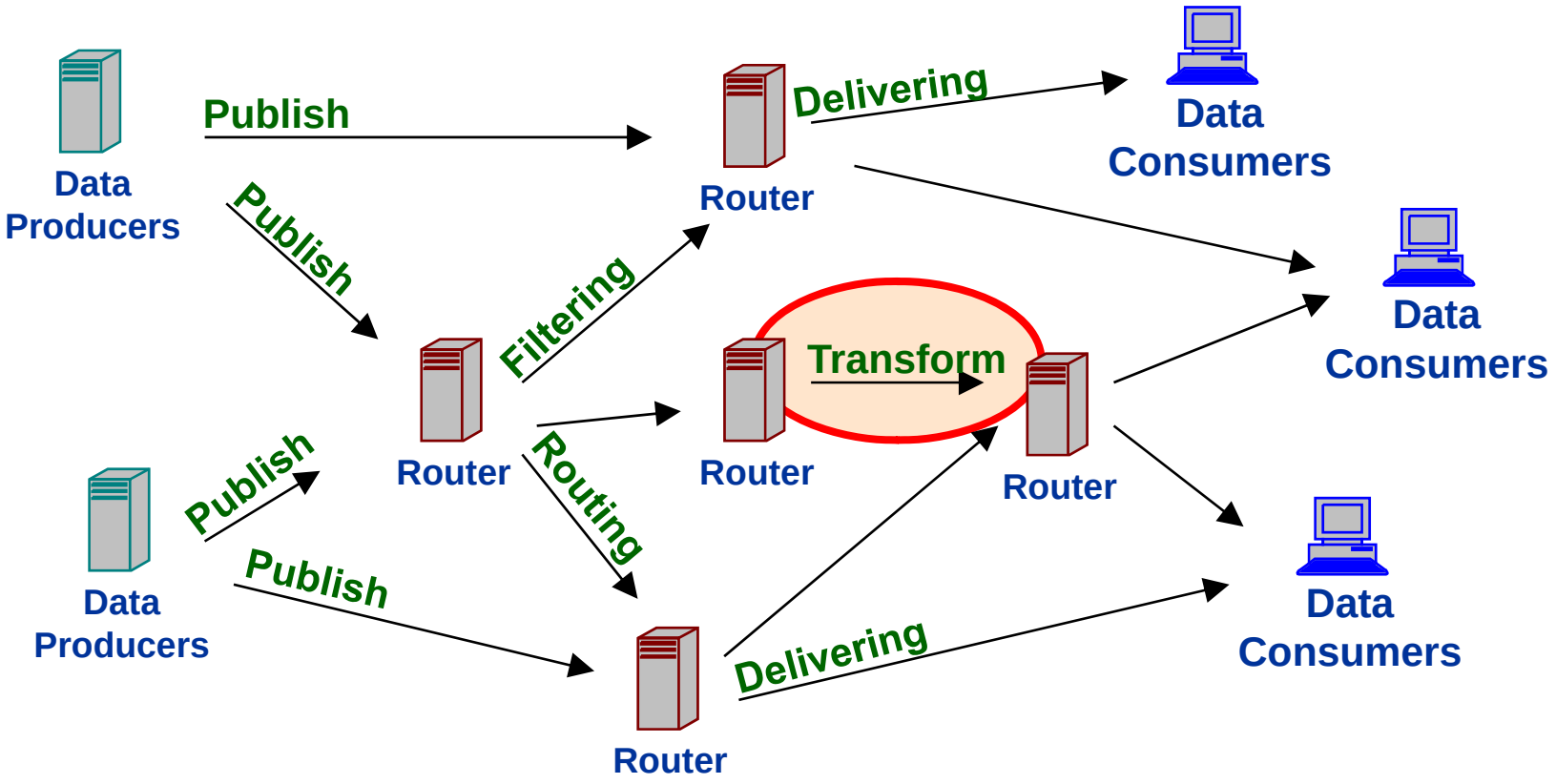


Publish/subscribe systems, monitor applications, content-routing systems ...

XML Stream Processing: Background

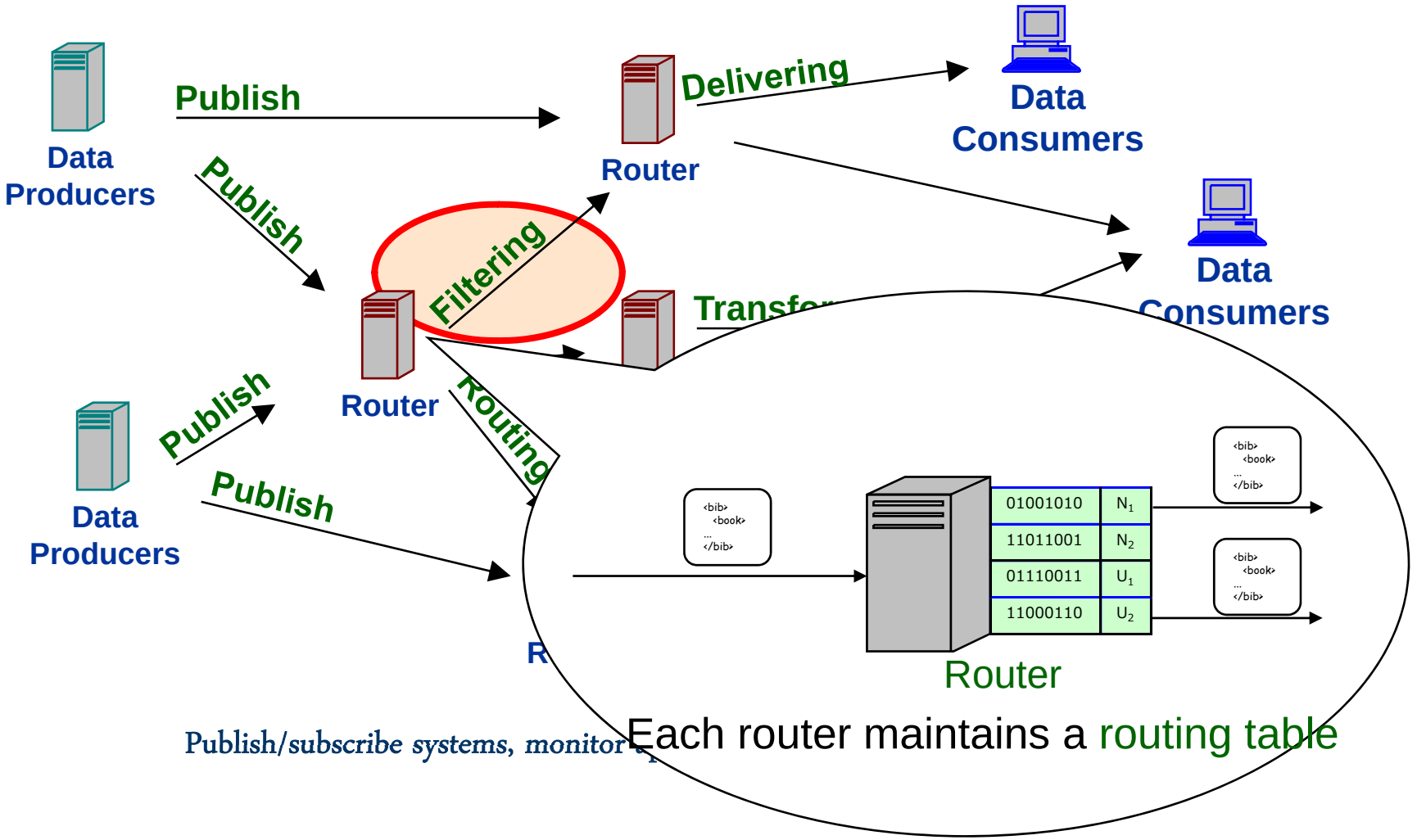
- XML stream: a sequence of XML packets, each of which is an XML document or a fragment of XML document
  - A large XML document is also treated as a stream of XML fragments (packets)
- Applications
  - Stock and sports tickers
  - Traffic information systems
  - Electronic personalized newspapers
  - …

XML Stream Processing: Application Scenario



Publish/subscribe systems, monitor applications, content-routing systems ...

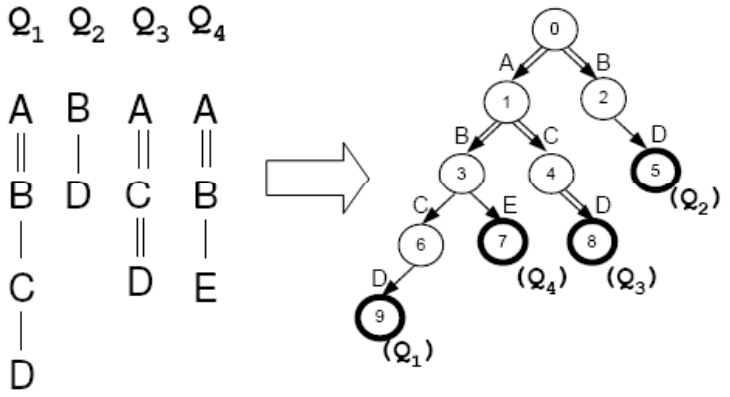
XML Stream Filtering: Application Scenario



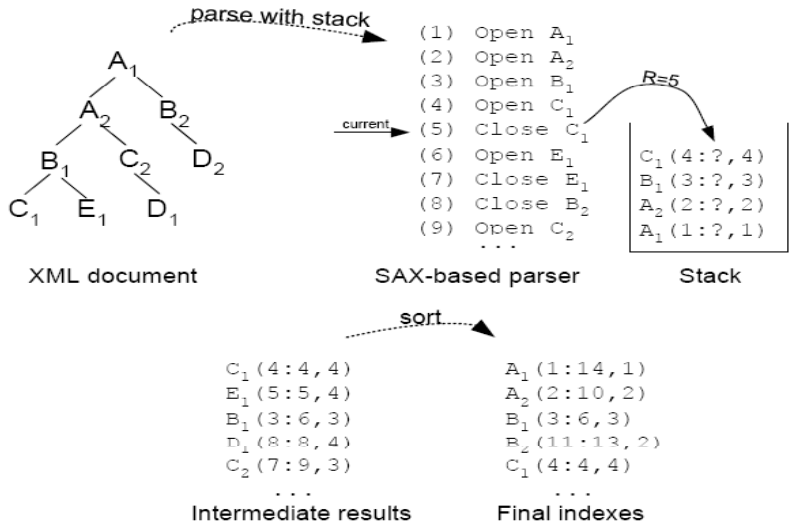
Publish/subscribe systems, monitor

XML Stream Filtering: Previous Solutions

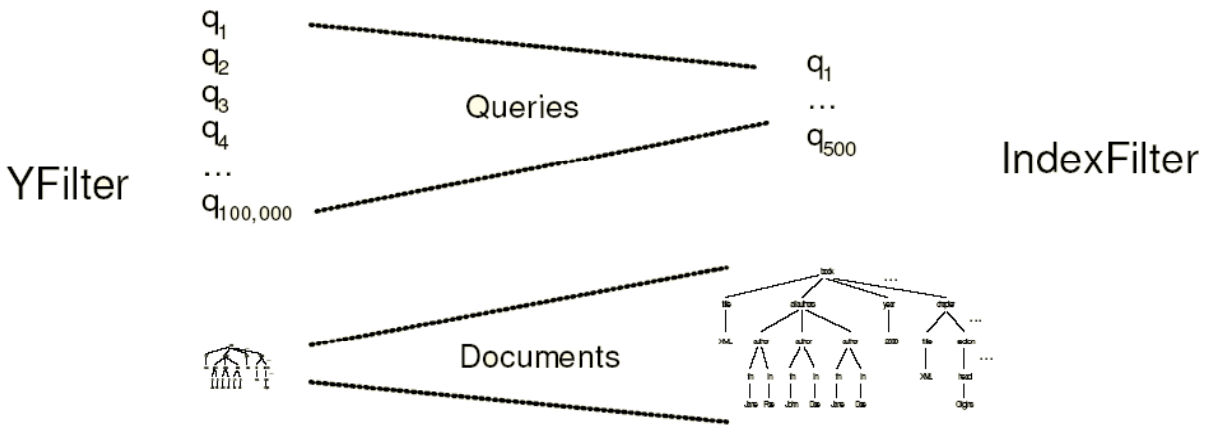
- Autonoma-based technology
  - Using an automata to build the routing table, which sharing the prefixes of all path queries.
  - XFilter[AF00], **YFilter**[DFF+02], XTrie[CFG+02], XPush[GS03], …



- Index-based technology
  - Building an (start, end, level) index on XML tags for each packet
  - Evaluate path prefix tree



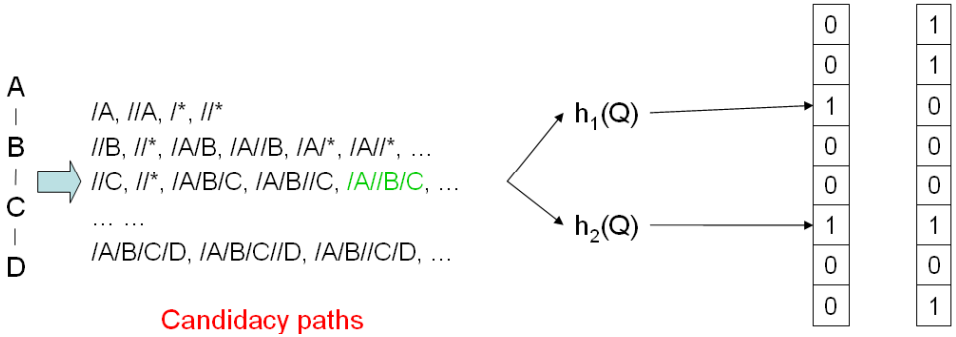
XML Stream Filtering: YFilter vs. IndexFilter

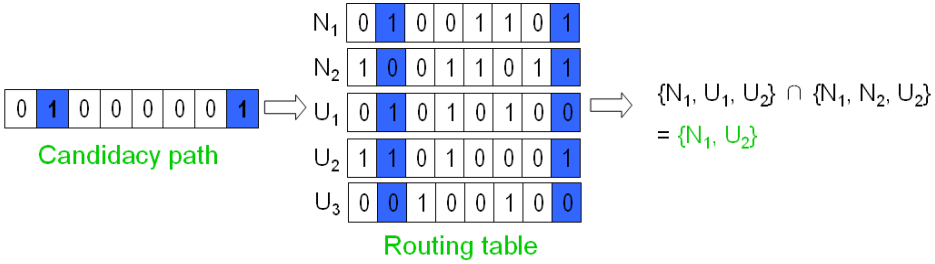


- YFilter is more efficient than Index-Filter when the number of queries is large and the XML packets are small.
- Index-Filter is fit for large XML documents and little queries.

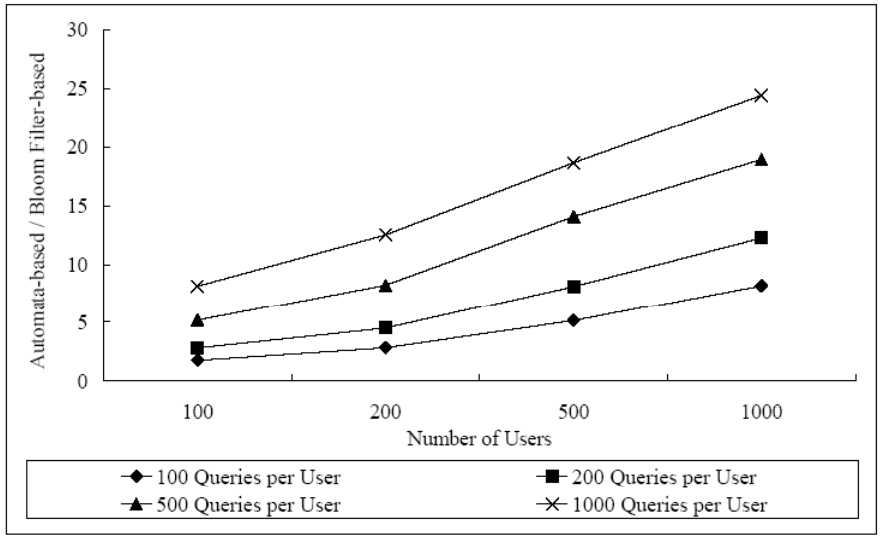
XML Stream Filtering: Bloom-filter based Method

- An XML path query is treated as a query string.
- Using Bloom Filter to build the routing table.
- An XML packet is parsed to generate a set of candidate paths.
- Prefix filters are used to decrease the number of redundant candidate paths.

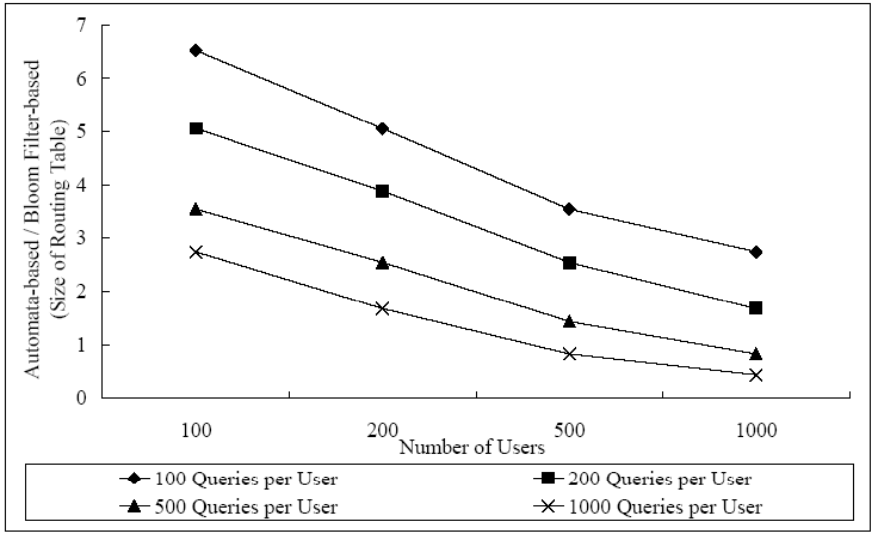




Routing Table Construction: Bloom Filter based vs. YFilter



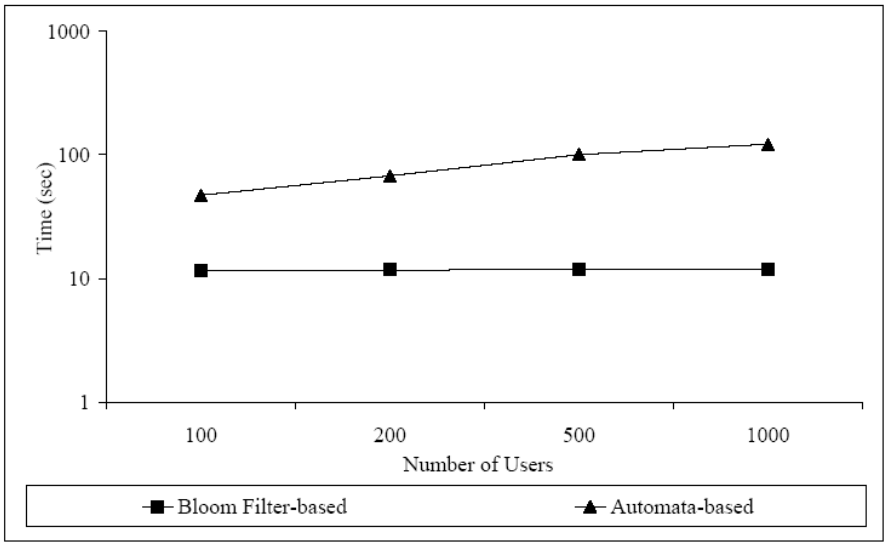
(a) Building Time



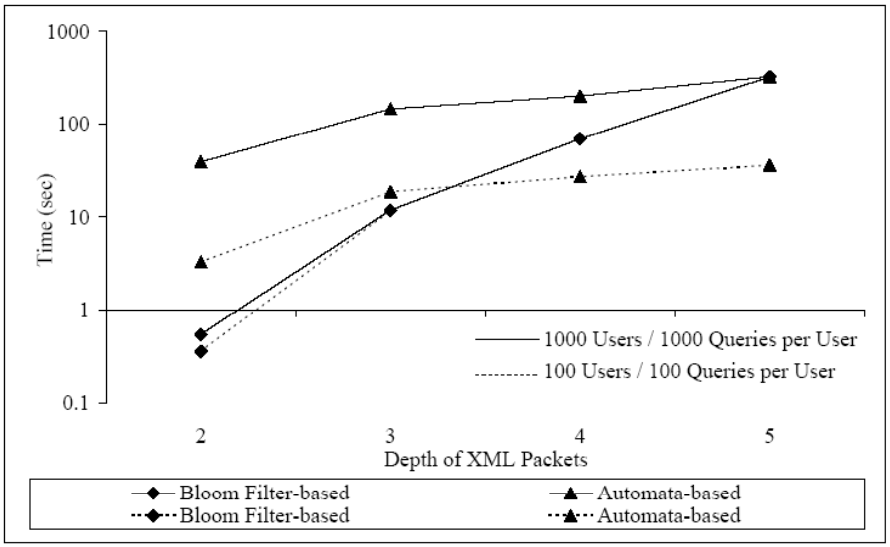
(b) Routing Table's Size

- ✓ **Bloom Filter-based is more efficient than Automata-based when building routing table**
- ✓ **The size of the routing table is much smaller when using Bloom Filter-based technique**

XML Packet Filtering: Bloom Filter based vs. YFilter



(a) 500 Queries per User



(b) 10,000 Queries vs. 1,000,000 Queries

- ✓ **Bloom Filter-based technique can handle millions of path queries efficiently.**
- ✓ **When the depth of the XML packet is no more 5, Bloom Filter-based is efficient than Automata-based.**

XML Stream Filtering: Summary

- Both high accuracy and high performance can be achieved by using Bloom-filter based approximate filtering method
- Current technique cannot efficiently process complex XML packets (i.e. XML documents with large depth)

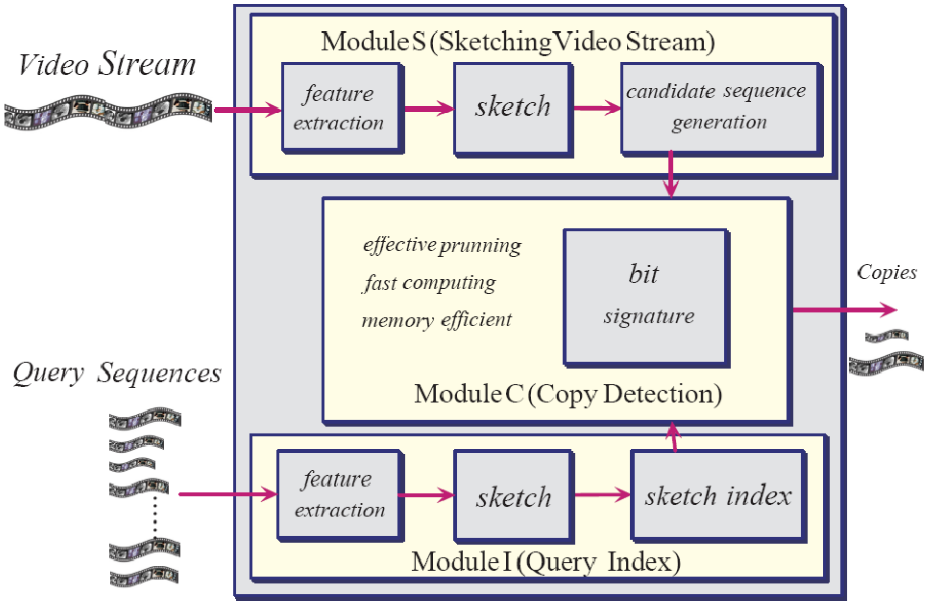
Video Stream

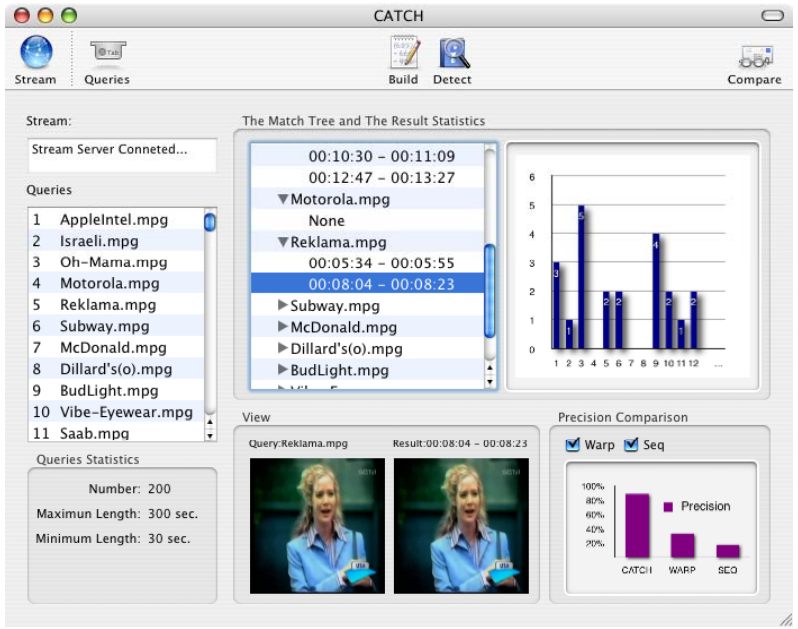
[Yan, ICDE08]

- Goal:
  - Continuous Content-Based Copy Detection over Streaming Videos
- Motivation
  - Copy right problem
    - Original videos may be edited with their frames being reordered, to avoid detection
  - VDBMS V.S. VDSMS
    - Many concurrent video streams and many continuous video copy monitoring queries.

Video Stream: System Implementation

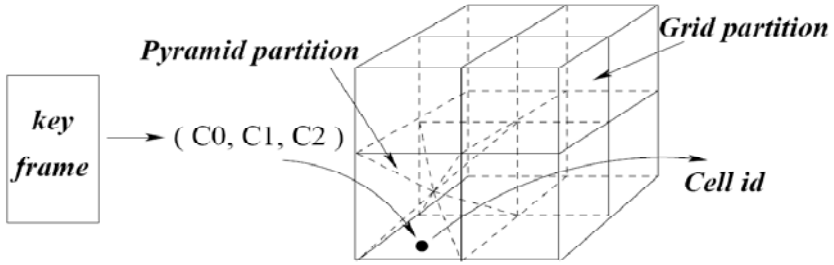
- Architecture





Video Stream: Key Techniques

- **Frame Fingerprint**
  - Frame signature extraction
  - Dimension reduction



- **Similarity Measure**

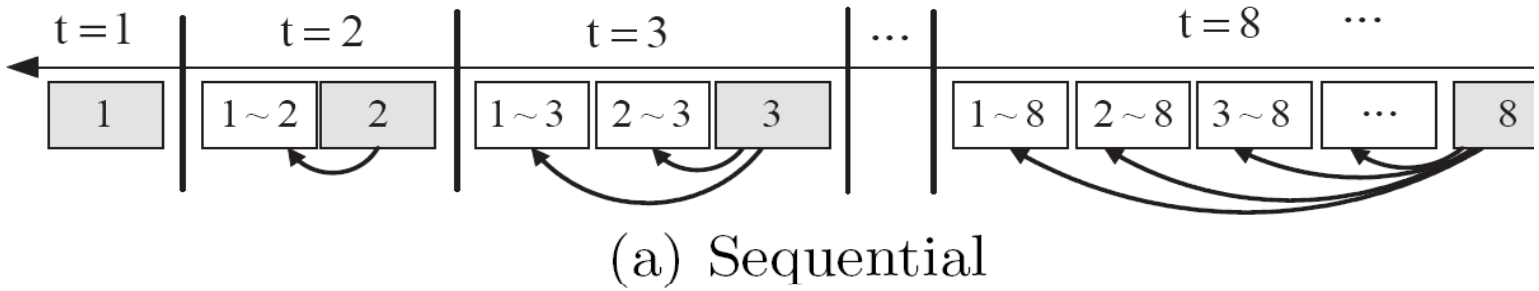
DEFINITION 2. *Video Sequence Similarity. Given two video sequences Q and P, the similarity is defined as:*

$$sim(Q,P) = \frac{|Q \cap P|}{|Q \cup P|}$$

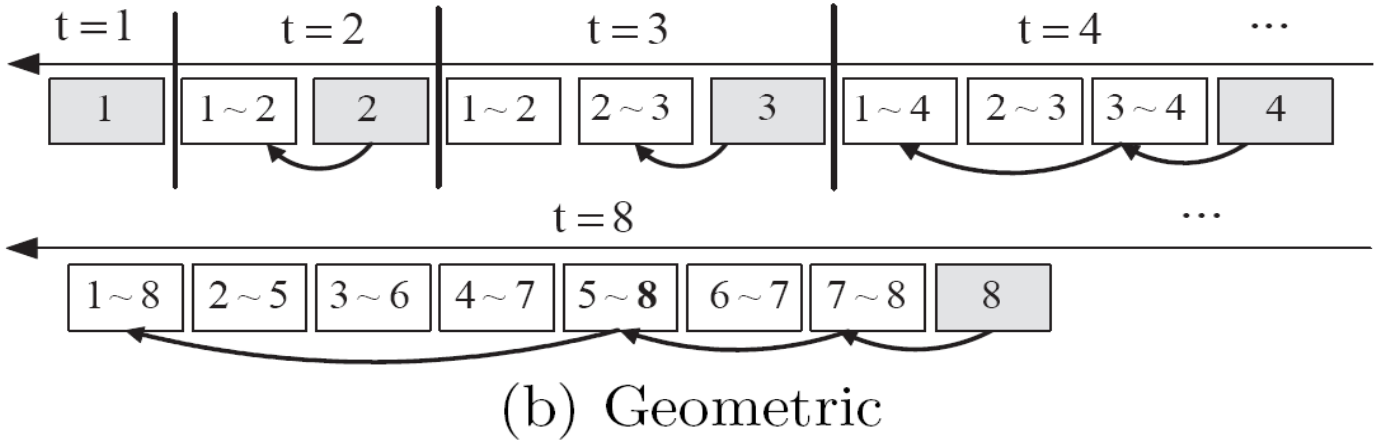
- **Mir**

$$Pr\{\tau(Q) = \tau(P)\} = \frac{|Q \cap P|}{|Q \cup P|}$$

Video Stream: Subsequences Comparison



(a) Sequential



(b) Geometric

Our Interests: Uncertainty

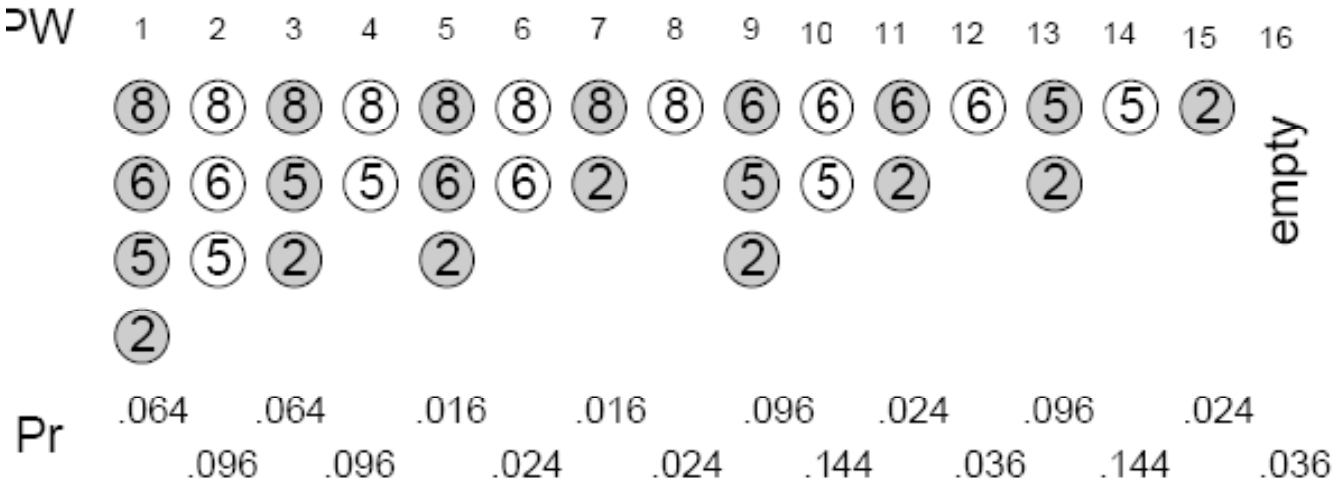
	2003	2004	2005	2006	2007	2008
Survey		JOS				
Frequent pattern	CIKM	VLDB	DASFAA, WAIM	IS		
Clustering, Density estimate	DASFAA			SDM,WAIM,JCST	KAIS,ICDE,JOS	
Stream system		JCRD	APWEB	DASFAA	FCSC	ICDE
Aggregate query			DASFAA	ICDE, JOS	ICDE	
Complex Data			ICDE			ICDE
Uncertainty						VLDB

Uncertain Stream: Definition

[Jin, VLDB08]

ID	Reading Info	Speed (×10)	prob.
1	AM 10:33, Honda, X-123	5	0.8
2	AM 10:35, Toyota, Y-245	6	0.5
3	AM 10:37, Mazda, Z-341	8	0.4
4	AM 10:38, Benz, W-541	2	0.4

Table 1: 4 Radar reading records



Framework

GOAL: designing a general framework for all kinds of top-k queries,
not only for a special kind of query.

1. A small subset of the original dataset
2. Self-maintenance $C(D \cup \{t\}) \subseteq C(D) \cup \{t\}$
3. Capable of answering a top-k query

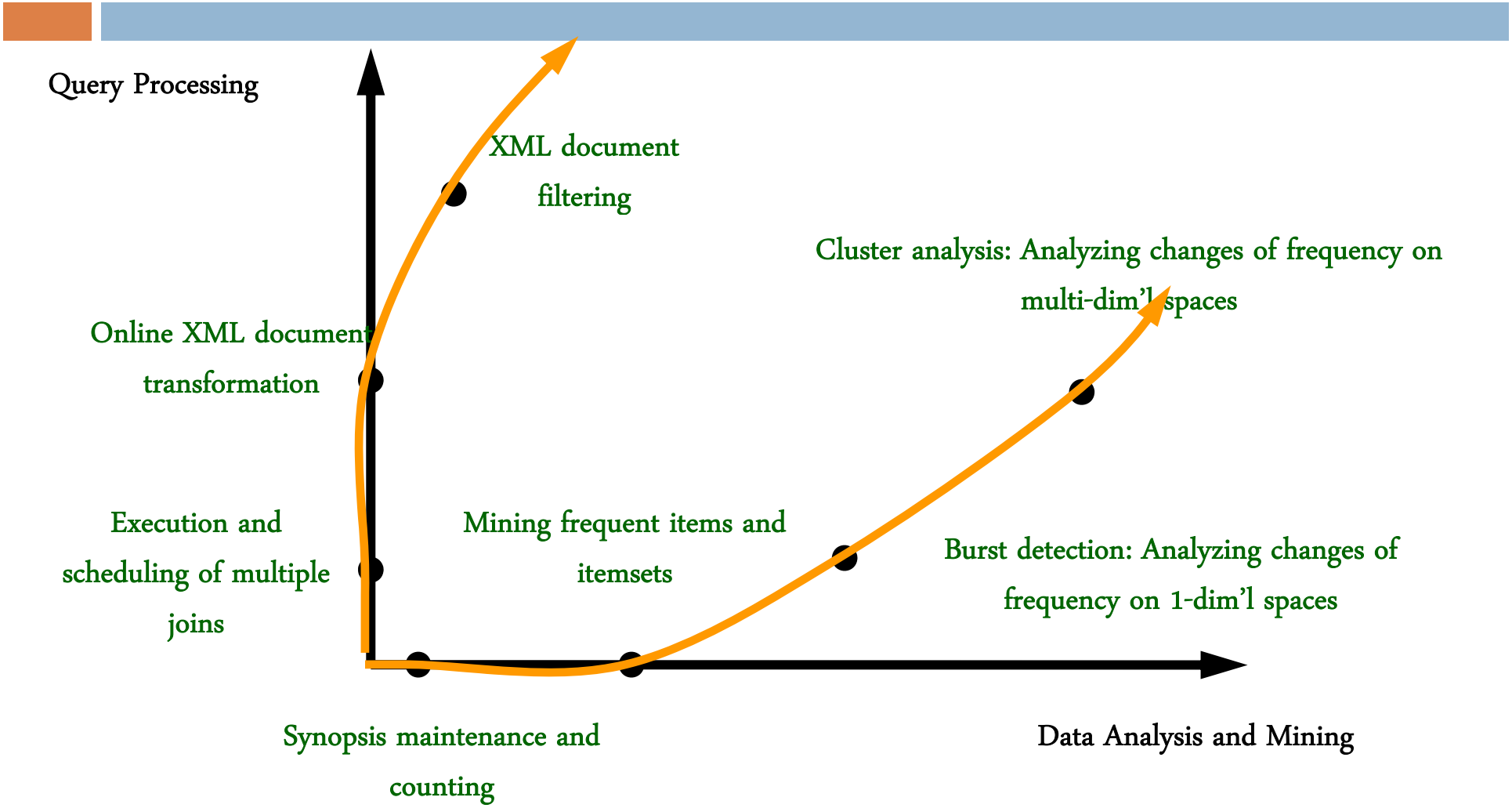
1. *W* different windows. i.e., [*t*-*j*, *t*], for *j*=0..*W*-1
2. One compact set for each window

1. CSQ, CCSQ, SCSQ, SCSQ-buffer
2. Space-efficient
3. Time-efficient

Performance summary

	Space consumption	Processing time
Basic Synopsis	$O(W + kH)$	$O(kH^2/W + \log W)$
Compact Set Queue	$O(H^2 \log W)$	$O(kH^2)$
Compressed Compact Set Queue	$O(H(k + \log W))$	$O(kH^2)$
Segmental Compact Set Queue	$O(H(k + \log W))$	$O(kH \log W)$
SCSQ-buffer	$O(H(k + \log W))$	$O(kH^2/W + \log W)$

Look ahead …


Query Processing
XML document filtering
Cluster analysis: Analyzing changes of frequency on multi-dim'l spaces
Online XML document transformation
Execution and scheduling of multiple joins
Mining frequent items and itemsets
Burst detection: Analyzing changes of frequency on 1-dim'l spaces
Synopsis maintenance and counting
Data Analysis and Mining

References

[AF00] Mehmet Altinel, Michael J. Franklin: Efficient Filtering of XML Documents for Selective Dissemination of Information, *VLDB'2000*

[AHWY03] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. *VLDB'*2003.

[AMS96] N. Alon, Y. Matias and M. Szegedy. The space complexity of approximating the frequency moments. STOC'1996

[BGK+03] Nicolas Bruno, Luis Gravano, Nick Koudas and Divesh Srivastava: Navigation- vs. Index-Based XML Multi-Query Processing. *ICDE'2003*

[CCFC02] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Proc. of the 29th ICALP*, 2002.

[CFG+02] C. Chan, P. Felber, M. Garofalakis, R. Rastogi: Efficient Filtering of XML Documents with XPath Expressions, *ICDE'2002*

[CM03] G. Cormode and S.Muthukrishnan. What's hot and what's not: Tracking most frequent items dynamically. In *Proc. of ACM PODS*, 2003.

[DFF+02] Yanlei Diao, Peter Fischer, Michael Franklin and Raymond To: YFilter: Efficient and Scalable Filtering of XML Documents. Demo paper. *ICDE'2002*

[GMMO00] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data stream. FOCS'2000

[GMO+03] Todd J. Green, Gerome Miklau, Makoto Onizuka, Dan Suciu: Processing XML Streams with Deterministic Automata, *ICDT 2003*

[GS03] Ashish Gupta and Dan Suciu: Stream Processing of XPath Queries with Predicates. *SIGMOD'2003*

[HFAE03] M. A. Hammad, M. J. Franklin and W. G. Aref and A. K. Elmagarmid. In proc of VLDB, 2003.

References (2003-2004)

- [Zhou, DASFAA03] Aoying Zhou, Zhiyuan Cai, Li Wei, Weining Qian: M-Kernel Merging: Towards Density Estimation over Data Streams. DASFAA, 2003: 285-292
- [Jin, CIKM03] Cheqing Jin, Weining Qian, Chaofeng Sha, Jeffrey X. Yu, and Aoying Zhou. Dynamically Maintaining Frequent Items over a Data Stream. CIKM, 2003
- [Jin, JOS04] Cheqing Jin, Weining Qian, and Aoying Zhou. The Analysis and Management of Streaming Data: A Survey. (In Chinese) Journal of Software (JOS), 15(8). 2004
- [Yan, JCRD04] Ying Yan, Cheqing Jin, Feng Cao, Hengjie Wang, Aoying Zhou, Load Shedding for Shared Window Joins over Data Streams, Journal of Computer Research and Development, (in Chinese), Vol. 41, No.10, 2004.
- [Yu, VLDB04] Jeffrey Xu Yu, Zhihong Chong, Hongjun Lu, and Aoying Zhou. False Positive or False Negative: Mining Frequent Itemsets from High Speed Transactional Data Streams. VLDB, 2004

References (2005)

- [Gong, ICDE05] Xueqing Gong, Weining Qian, Ying Yan and Aoying Zhou: Bloom Filter-based XML Packets Filtering for Millions of Path Queries. ICDE'2005
- [Zhou, DASFAA05] Aoying Zhou, Shouke Qin, Weining Qian: Adaptively Detecting Aggregation Bursts in Data Streams. DASFAA 2005: 435-446
- [Chong, DASFAA05] Zhihong Chong, Jeffrey Xu Yu, Hongjun Lu, Zhengjie Zhang, Aoying Zhou: False-Negative Frequent Items Mining from Data Streams with Bursting. DASFAA 2005: 422-434
- [Xia, WAIM05] Tian Xia, Cheqing Jin, Xiaofang Zhou, Aoying Zhou: Filtering Duplicate Items over Distributed Data Streams. WAIM 2005: 779-784
- [Jin, APWEB05] Cheqing Jin, Aoying Zhou: Distinct Estimate of Set Expressions over Sliding Windows. APWeb 2005: 530-535

References (2006)

- [Qin, ICDE06] Shouke Qin, Weining Qian, Aoying Zhou: Approximately Processing Multi-granularity Aggregate Queries over Data Streams. ICDE 2006
- [Yu, IS06] Jeffrey Xu Yu, Zhihong Chong, Hongjun Lu, Zhenjie Zhang, Aoying Zhou: A false negative approach to mining frequent itemsets from high speed transactional data streams. Inf. Sci. 176(14): 1986-2015 (2006)
- [Cao, SDM06] Feng Cao, Martin Ester, Weining Qian, Aoying Zhou: Density-Based Clustering over an Evolving Data Stream with Noise. SDM 2006
- [Chong, JCST06] Zhihong Chong, Jeffrey Xu Yu, Zhengjie Zhang, Xuemin Lin, Wei Wang, Aoying Zhou: Efficient Computation of k-Medians over Data Streams Under Memory Constraints. J. Comput. Sci. Technol. 21(2): 284-296 (2006)
- [Zhou, DASFAA06] Yongluan Zhou, Ying Yan, Feng Yu, Aoying Zhou: PMJoin: Optimizing Distributed Multi-way Stream Joins by Stream Partitioning. DASFAA 2006: 325-341
- [Cao, WAIM06] Feng Cao, Anthony K. H. Tung, Aoying Zhou: Scalable Clustering Using Graphics Processors. WAIM 2006: 372-384
- [Qin, JOS06] Shouke Qin, Weining Qian, Aoying Zhou. Fractal-Based Algorithms for Burst Detection over Data Streams. Journal of Software. Vol. 17, No. 9, 2006. (In Chinese)

References (2007)

- [Zhou, KAIS07] Aoying Zhou, Feng Cao, Weining Qian and Cheqing Jin. Tracking clusters in evolving data streams over sliding windows. In Knowledge and Information Systems: An International Journal(KAIS Journal), Vol. 10, No. 3, March, 2007.
- [Zhou, ICDE07] Aoying Zhou, Feng Cao, Ying Yan, Chaofeng Sha, Xiaofeng He: Distributed Data Stream Clustering: A Fast EM-based Approach. ICDE 2007: 736-745
- [Böhm, ICDE07] Christian Böhm, Beng Chin Ooi, Claudia Plant, Ying Yan: Efficiently Processing Continuous k-NN Queries on Data Streams. ICDE 2007: 156-165
- [Jin, FCST07] Cheqing Jin, Aoying Zhou, Jeffrey Xu Yu, Joshua Zhexue Huang and Feng Cao. Adaptive scheduling for shared window joins over data streams. Frontiers of Computer Science in China, Vol. 01, No. 04, 2007
- [Chang, JOS07] Jianlong Chang, Feng Cao, Aoying Zhou. Clustering evolving data streams over sliding windows. Journal of Software, Vol. 18, No. 4, 2007 (In Chinese)

References (2008)

- [Yan, ICDE08] Ying Yan, Beng Chin Ooi, Aoying Zhou: Continuous Content-Based Copy Detection over Streaming Videos. ICDE 2008: 853-862
- [Zhou, ICDE08] Aoying Zhou, Ying Yan, Xueqing Gong, Jianlong Chang, Dai Dai: SMART: A System for Online Monitoring Large Volumes of Network Traffic. ICDE 2008: 1576-1579 (demo)
- [Jin, VLDB08] Cheqing Jin, Ke Yi, Lei Chen, Jeffrey Xu Yu, Xuemin Lin. Sliding-window top-k queries on Uncertain Streams. VLDB, 2008

Thanks for Your Attention!

Q&A