Top-K Query Processing Techniques for Distributed Environments

by

Demetris Zeinalipour

Visiting Lecturer
Department of Computer Science
University of Cyprus

Wednesday, June 7th, 2006
"Mediteranean Studies" Seminar Room,
FORTH, Heraklion, Crete

http://www.cs.ucy.ac.cy/~dzeina/

Presentation Goals

- To provide an **overview** of **Top-K Query Processing algorithms** for centralized and distributed settings.

- To present the **Threshold Join Algorithm (TJA)** which is our distributed top-k query processing algorithm.

- To present other **research activities** that are directly or indirectly related to this work.

Data Management & Query Processing Today

We are living in a world where data is generated All The Time & Everywhere

Characteristics of these Applications

- ***"Data is generated in a distributed fashion"*** *e.g. sensor data, file-sharing data, Geographically Distributed Clusters)*

- ***"Distributed Data is often outdated before it is ever utilized"***

  (e.g. CCTV video traces, Internet ping data, sensor readings, weblogs, RFID Tags,…)

- ***"Transferring the Data to a centralized repository is usually more expensive than storing it locally"***

Motivating Question

- ***Why design algorithms and systems that a' priori organize information in centralized repositories?***

- **Our Approach: *"In-situ Data Storage & Retrieval"***
  - *Data remains **in-situ** (at the generating site).*
  - *When Users want to search/retrieve some information they perform **on-demand** queries.*

- ***Challenges:***
  - ***Minimize the utilization** of the communication medium*
  - ***Exploit the network** and the **inherent parallelism of a distributed environment.** Focus on Hierarchical Networks are ubiquitous (e.g. P2P, and sensor-nets).*
  - *Number of Answers might be very large* ➔ ***Focus on Top-K***

Presentation Outline

1. **Introduction to Top-K Query Processing**
2. Related Work & Algorithms
3. The Threshold Join Algorithm (TJA)
4. Experimental Evaluation using our Middleware Testbed.
5. Related Activities & Future Work.

Distributed Top-K Query Processing

TOP-k Query Objectives:

1. To find the **k highest ranked** answers to a user defined scoring function

   (e.g. Record1: 0.7 red, Record2: 0.4 red, etc)

2. Minimize some **cost metric** associated with the retrieval of the complete answer set.

Distributed Top-K Query Processing

Cost Metric in a Distributed Environment

A) Bandwidth

- Transmitting less data conserves resources, energy and minimizes failures.
- e.g. in a Sensor Network sending 1 byte ≈ 1120 CPU instructions.

Source: The RISE (Riverside Sensor) (NetDB'05, IPSN'05 Demo, IEEE SECON'05)

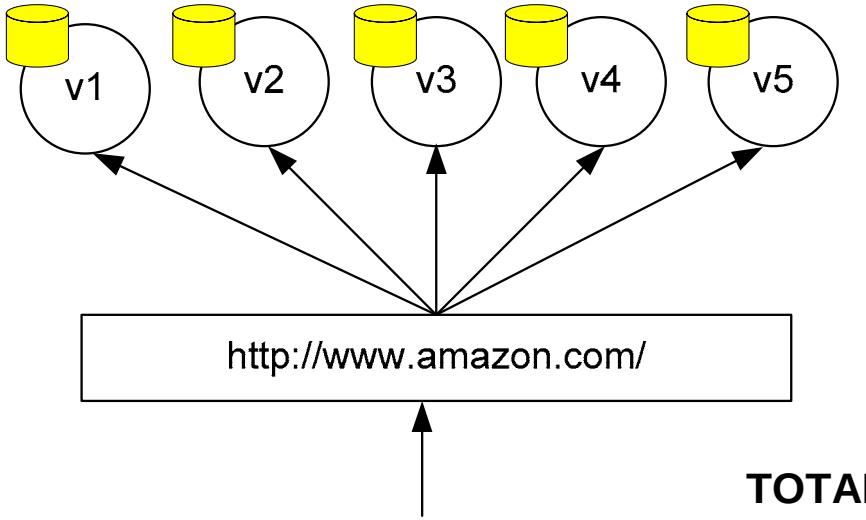
B) Query Response Time

- The #bytes transmitted is not the only parameter.
- We want to minimize the time to execute a query.

Distributed Top-K Query Processing

Motivating Example

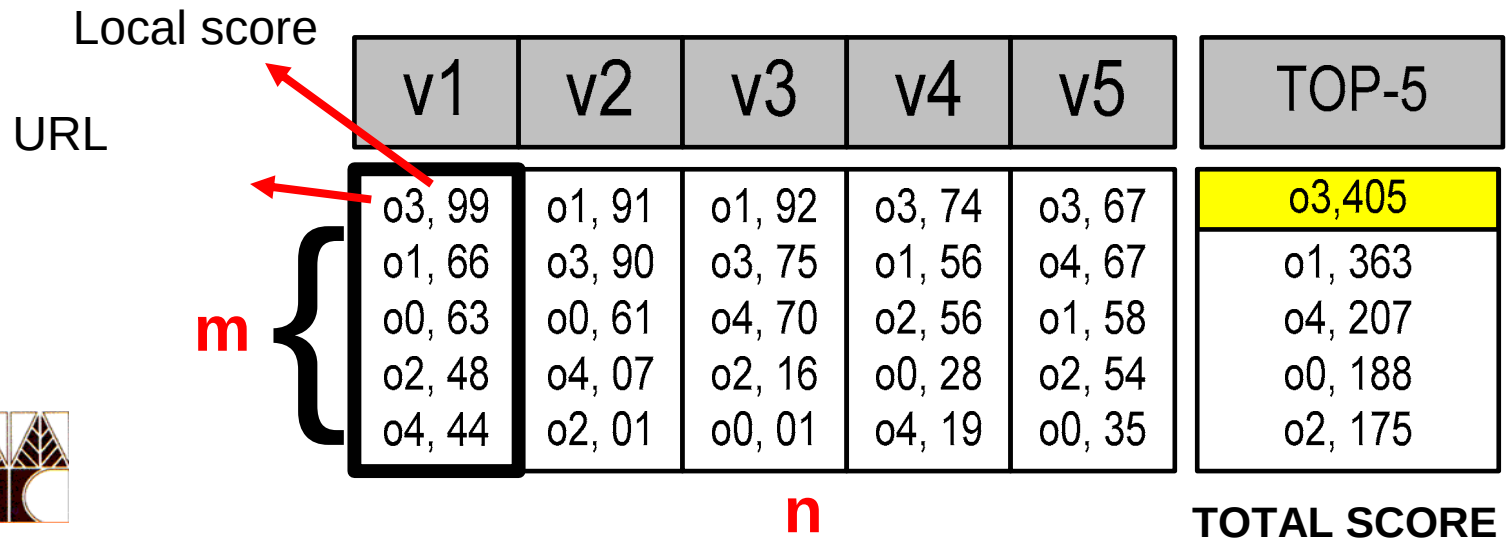
- Assume that we have a cluster of **n=5 webservers.**
- Each server maintains locally the same **m=5 webpages.**
- When a web page is accessed by a client, a server increases a local **hit counter** by one.



Distributed Top-K Query Processing

Motivating Example (cont'd)

- ***TOP-1 Query:*** *"Which Webpage has the highest number of hits across all servers (i.e. highest* ***Score(o_i))****?"*
- ***Score(o_i)*** *can only be calculated if we combine the hit count from all 5 servers.*



Distributed Top-K Query Processing

Other Applications

- **Sensor Networks**: Each sensor maintains locally a sliding window of the last **m readings (i.e. m (ts, val) pairs).**

  Q: *Find when did we have the K=3 highest average temperatures across all sensors.*

- **Other Applications:** Collaborative Spam Detection Networks, Content Distribution Networks, Information Retrieval, etc

Presentation Outline

1. **Introduction to Top-K Query Processing**

2. **Related Work & Algorithms**

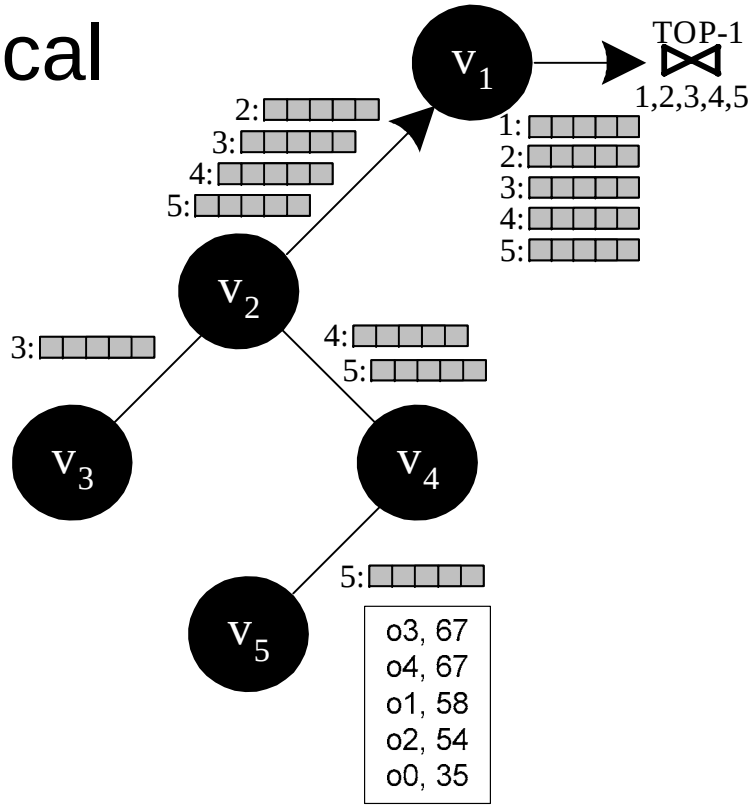
3. The Threshold Join Algorithm (TJA)

4. Experimental Evaluation using our Middleware Testbed.

5. Related Activities & Future Work.

Naïve Solution: **Centralized Join (CJA)**

- Each Node sends all its local scores (list)
- Each intermediate node forwards all received lists
- The Gnutella Approach



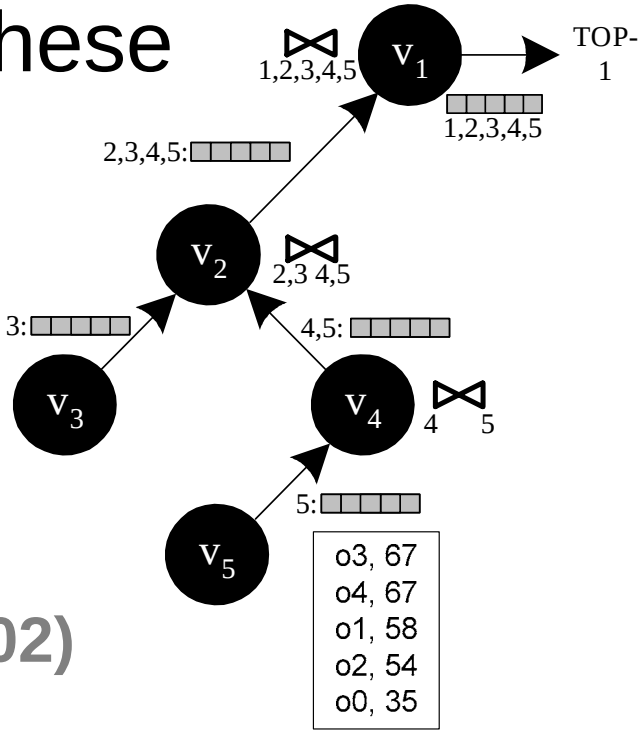
Drawbacks

- Overwhelming amount of messages.
- Huge Query Response Time

Improved Solution: **Staged Join (SJA)**

- Aggregate the lists before these are forwarded to the parent using:

$$R(v_i) = list(v_i) \bowtie \left(\bowtie_{\forall j \in children(v_i)} list(v_j) \right)$$



- This is essentially the TAG approach **(Madden et al. OSDI '02)**

- **Advantage:** Only (n-1) messages

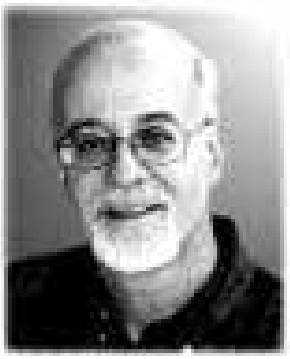
- **Drawback:** Still sending everything!

The **Threshold Algorithm (Not Distributed)**

Fagin's* Threshold Algorithm (TA):

Long studied and well understood.

* Concurrently developed by 3 groups



TA Algorithm

v1	v2	v3	v4	v5
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35

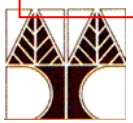
1) Access the ***n*** lists in parallel.

2) While some object o_i is seen, perform a **random access** to the other lists to find the complete score for o_i.

3) Do the same for all objects in the current row.

4) Now compute the threshold τ as the **sum of scores** in the current row.

5)The algorithm stops after ***K*** objects have been found with a score above τ.



The **Threshold Algorithm (Example)**

v1	v2	v3	v4	v5	TOP-K
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67	O3, 405
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67	O1, 363
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58	O4, 207
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54	
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35	

Iteration 1 Threshold

τ = 99 + 91 + 92 + 74 + 67 => **τ = 423**

Have we found K=1 objects with a score above τ? => NO

Iteration 2 Threshold

τ (2nd row)= 66 + 90 + 75 + 56 + 67 => **τ = 354**

Have we found K=1 objects with a score above τ? => YES!

The **Threshold Algorithm (Not Distributed)**

Why is the threshold correct?

Because the threshold essentially gives us the maximum Score for the objects not seen (<= τ)

Advantages:

- The number of object accessed is minimized!

Why Not TA in a distributed Environment?

Disadvantages:

Each object is accessed individually (random accesses)

➔A **huge** number of **round trips** (phases)

➔**Unpredictable Latency** (Phases are sequential)

➔In network **Aggregation not possible**

Presentation Outline

1. **Introduction to Top-K Query Processing**

2. **Related Work & Algorithms**

3. **The Threshold Join Algorithm (TJA)**

4. Experimental Evaluation using our Middleware Testbed.

5. Related Activities & Future Work.

Threshold Join Algorithm (TJA)

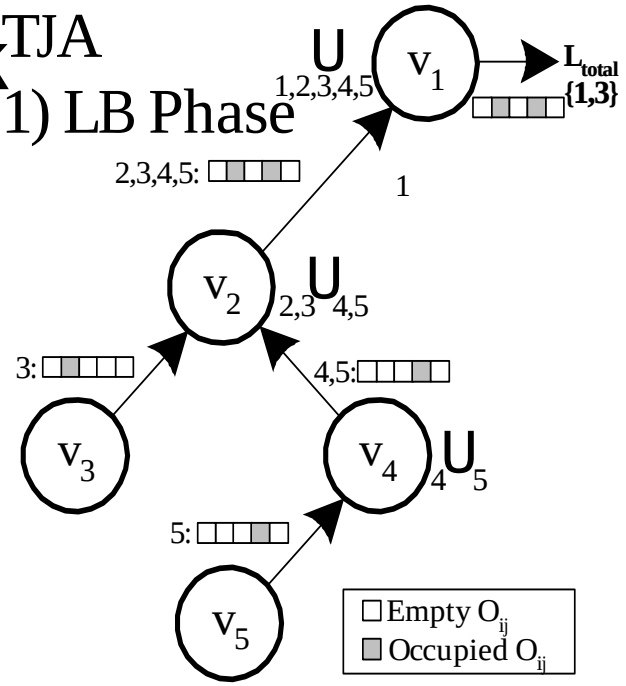
- TJA is our **3-phase algorithm** that **minimizes** the number of transmitted objects and hence the **utilization of the communication channel**.

- **How does it work:**
  1. **LB Phase:** Ask each node to send the K (locally) highest ranked results.

     The union of these results defines a threshold τ.
  2. **HJ Phase:** Ask each node to transmit everything above this threshold τ.
  3. **CL Phase:** If at the end we have not identified the complete score of the K highest ranked objects, then we perform a cleanup phase to identify the complete score of all incompletely calculated scores.

Step 1 - LB (Lower Bound) Phase

- Each node sends its **top-k** results to its parent.
- Each intermediate node performs a **union** of all received lists (denoted as **τ**):

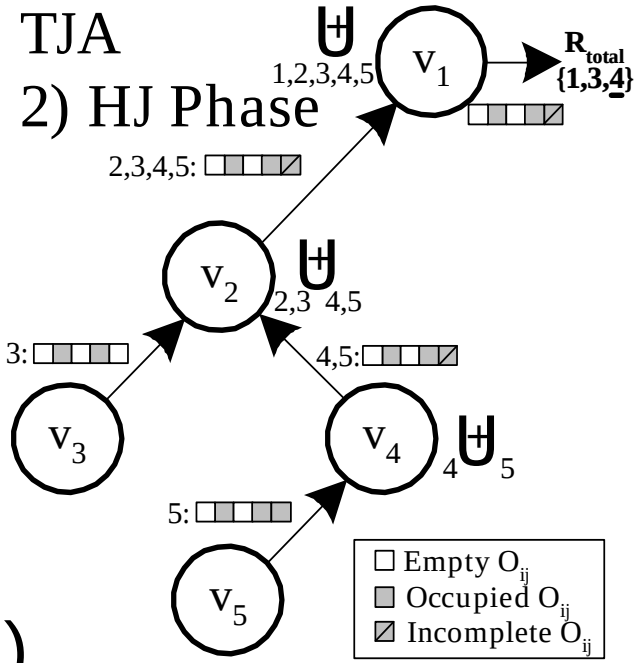


Query: TOP-1

v1	v2	v3	v4	v5	LB
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67	{o3, o1}
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67	
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58	
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54	
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35	

Step 2 – HJ (Hierarchical Join) Phase

- Disseminate τ to all nodes
- Each node sends back everything with score above all objectIDs in τ.
- Before sending the objects, each node tags as **incomplete,** scores that could not be computed exactly (upper bound)



v1	v2	v3	v4	v5	HJ
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67	o3, 405
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67	o1, 363
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58	o4',354
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54	
o4, 44	o2, 01	o0, 01	o4,19	o0, 35	

Complete

Incomplete

Step 3 – CL (Cleanup) Phase

Have we found K objects with a complete score?

Yes: The answer has been found!

No: Find the *complete score* for each incomplete object (all in a single batch phase)

- **CL ensures correctness!**
- **This phase is rarely required in practice.**

v1	v2	v3	v4	v5
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35

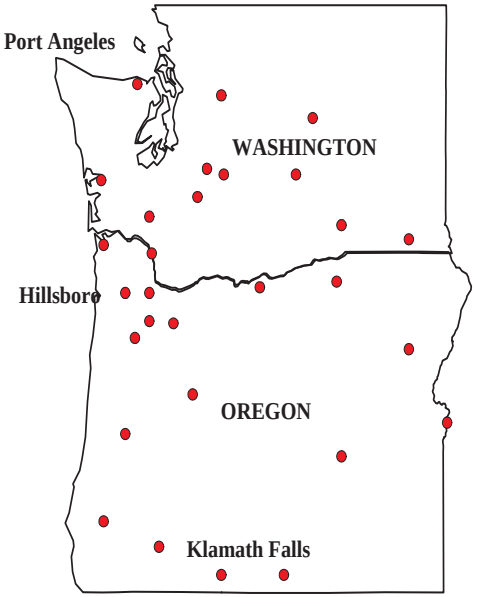
TOP-5
o3,405
o1, 363
o4, 207
o0, 188
o2, 175

Presentation Outline

1. **Introduction to Top-K Query Processing**
2. Related Work & Algorithms
3. The Threshold Join Algorithm (TJA)
4. **Experimental Evaluation using our Middleware Testbed.**
5. Conclusions & Future Work.

Experimental Evaluation

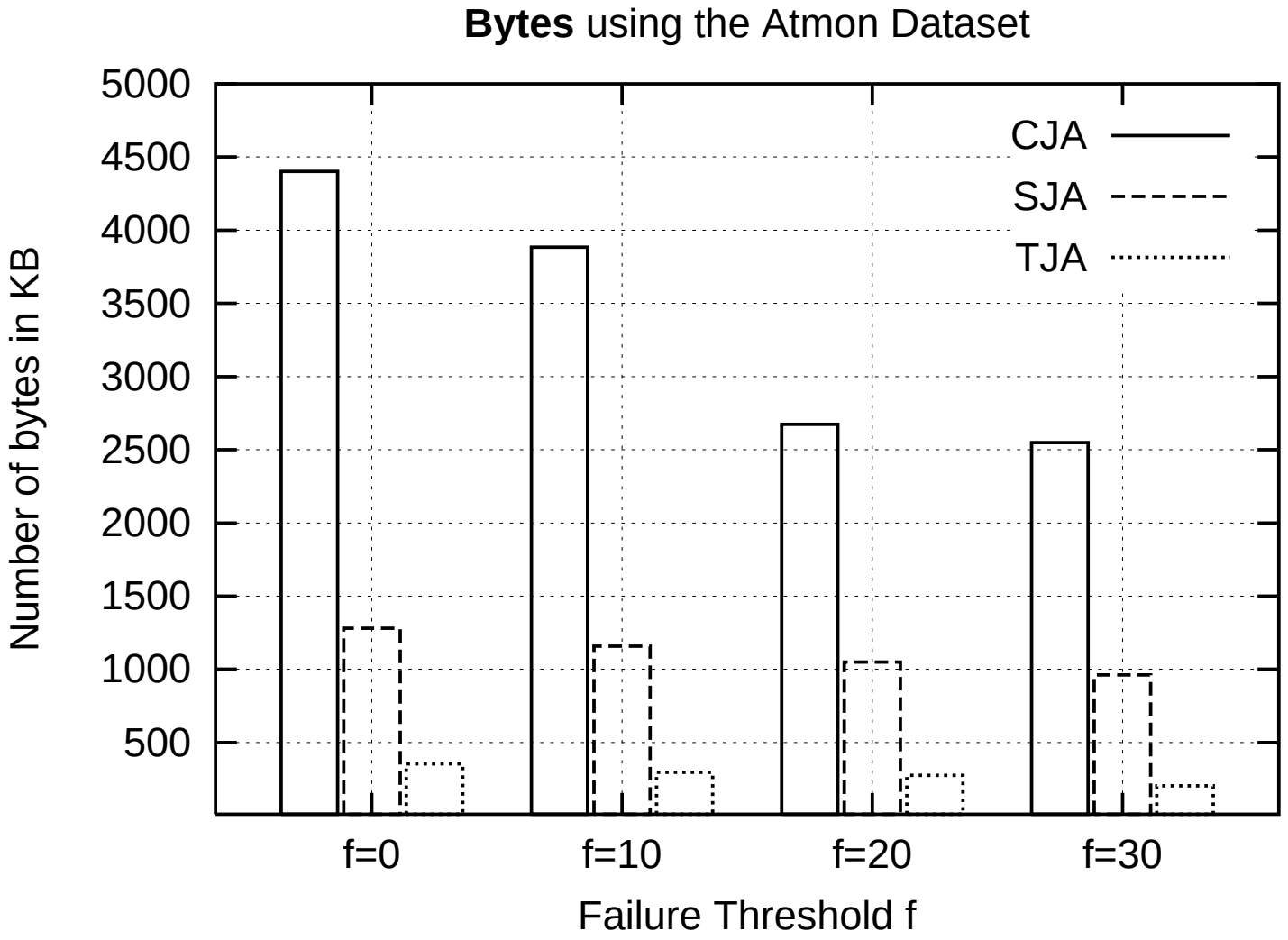
- We implemented a real P2P middleware in JAVA (sockets + binary transfer protocol).
- We tested our implementation with a network of 1000 real nodes using 75 Linux workstations.
- We use a trace driven experimentation methodology.



For the results presented in this talk:

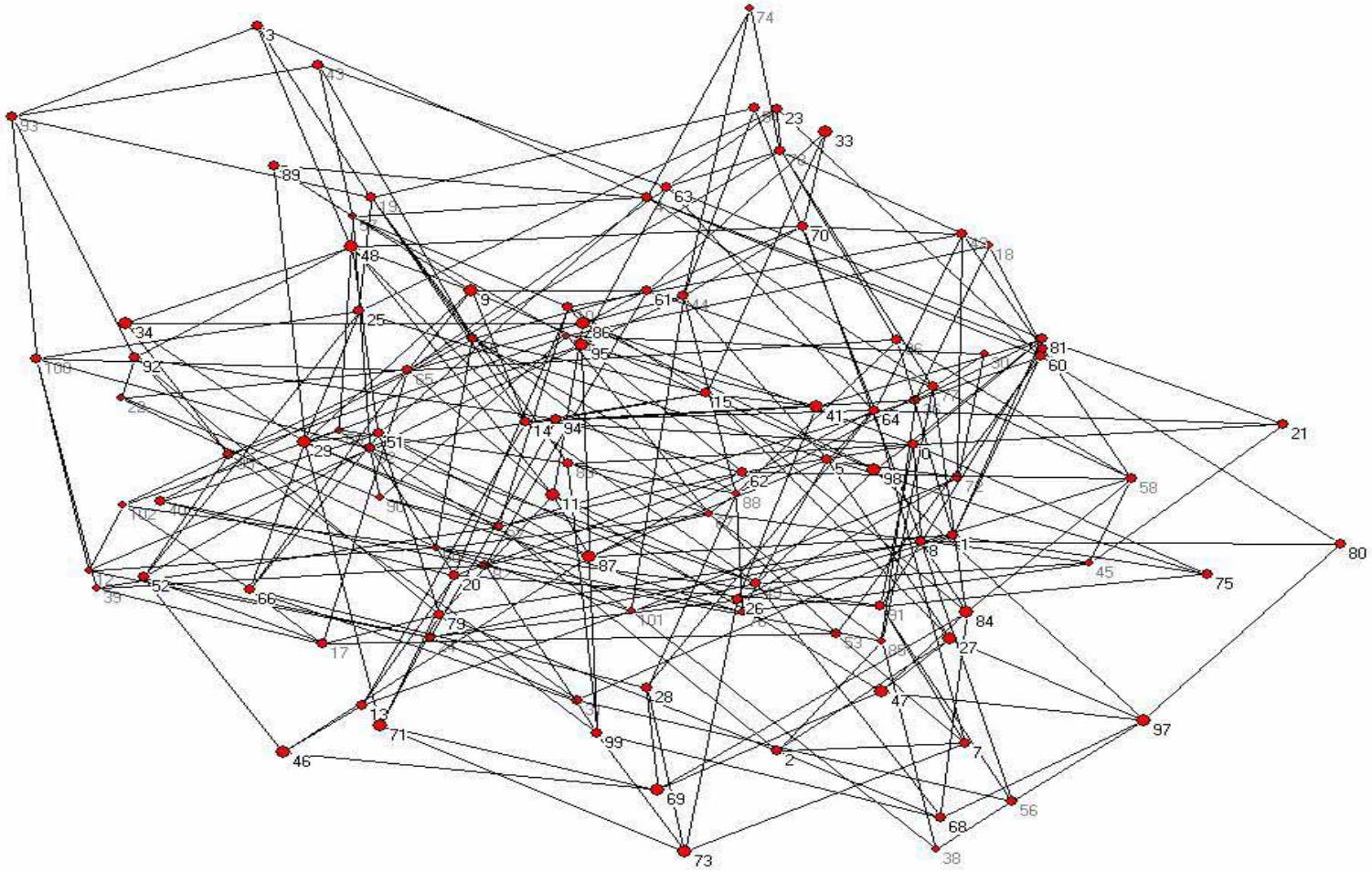
- **Dataset:** Environmental Measurements from 32 atmospheric monitoring stations in Washington & Oregon. (2003 – 2004)
- **Query:** K timestamps on which average temperature across all stations was maximum
- **Network:** Random Graph (degree=4, diameter 10)
- **Evaluation Criteria:** i) **Bytes**, ii) **Time**, **iii) Messages**

Experimental Results



TJA requires one order of magnitude less bytes than the Centralized Algorithm!

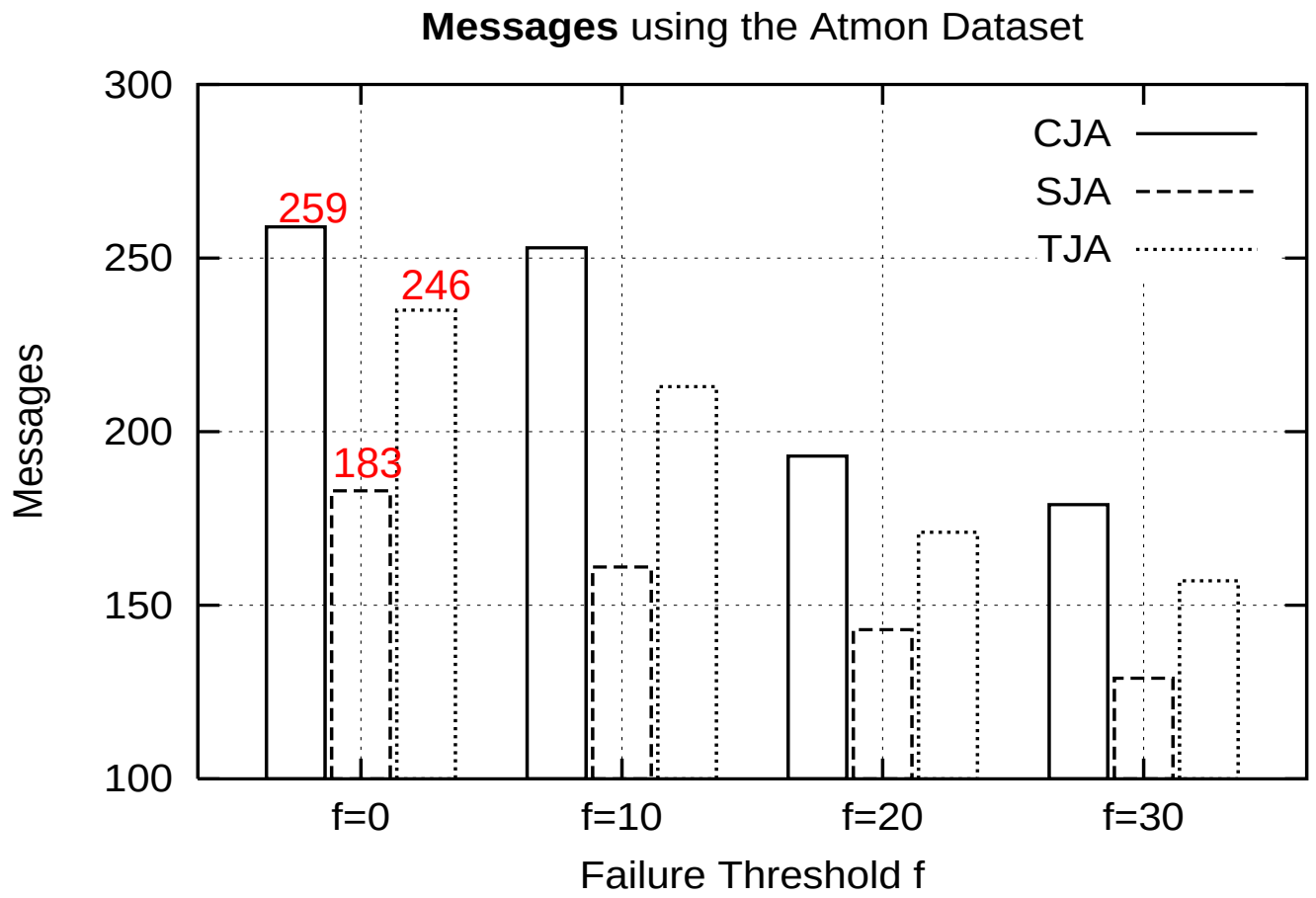
Experimental Results



TJA: 3,797ms [LB:1059ms, HJ:2730ms, CL:8ms]

SJA: 8,224ms **CJA:18,660ms**

Experimental Results



Although TJA consumes more messages than SJA, these are small size messages

The TPUT Algorithm

v1	v2	v3	v4	v5	TOP-1
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67	o1=183, o3=240
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67	o3=405
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58	o1=363
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54	o2'=158
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35	o4'=137
					o0'=124

P1 P2 P3

Q: TOP-1

Phase 1 : **o1** = 91+92 = 183, **o3** = 99+67+74 = 240

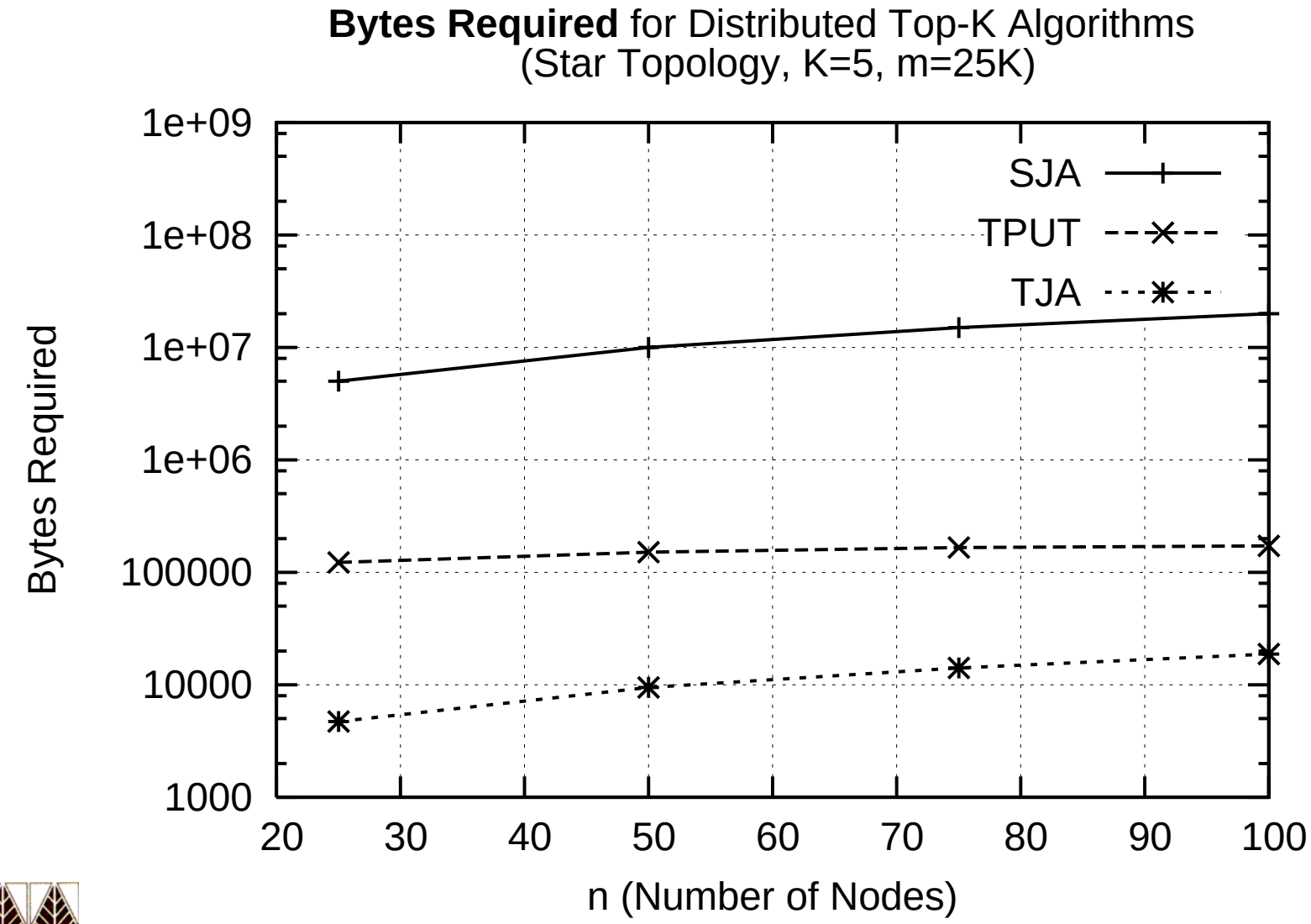
τ = (Kth highest score (partial) / n) => 240 / 5 => τ = 48

Phase 2 : **Have we computed K exact scores ?**

Computed Exactly: [o3, o1] Incompletely Computed: [o4,o2,o0]

Drawback: The threshold is too coarse (uniform)

TJA vs. TPUT


Bytes Required for Distributed Top-K Algorithms
(Star Topology, K=5, m=25K)
Bytes Required
1e+09
1e+08
1e+07
1e+06
100000
10000
1000
20
30
40
50
60
70
80
90
100
n (Number of Nodes)
SJA
TPUT
TJA

Presentation Outline

1. **Introduction to Top-K Query Processing**
2. Related Work & Algorithms
3. The Threshold Join Algorithm (TJA)
4. Experimental Evaluation using our Middleware Testbed.
5. **Conclusions & Future Work.**

Conclusions

- Distributed Top-K Query Processing is a new area with many **new challenges** and **opportunities**!

- We showed that the TJA is an efficient algorithm for computing the K highest ranked answers in a distributed environment.

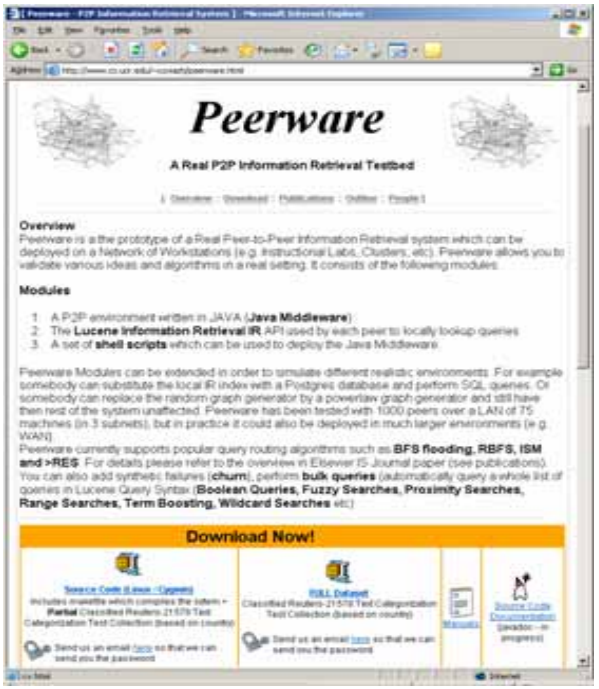
- We believe that our algorithm will be a useful component in Query Optimization engines of future Database systems.

Future Work

- Implementation of the TJA algorithm in nesC the programming language of TinyOS. Deployment using the Riverside Sensor

- Provide the implementation of TJA as an extension of our Open Source P2P Information Retrieval Engine :

  http://www.cs.ucr.edu/~csyiazti/peerware.html



- Explore other domains in which the discussed ideas might be beneficial Grids, vehicular networks, etc.

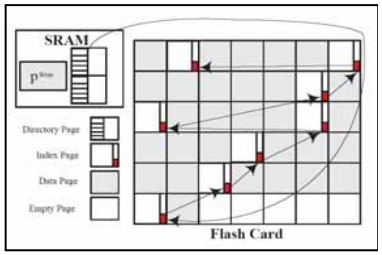
Related Activity 1: Sensor Local Access Methods

- TJA assumes that **random** and **sequential** access methods to local data is available at each site.

RISE Sensor

- **Problem: What happens if the target device is a battery-limited sensor device?**

- Distinct Characteristics
  - New storage medium: FLASH memory
  - Asymmetric Read/Write Characteristics



- We propose ***"MicroHash: An Efficient Index Structure for Flash-Based Sensor Devices",***

  D. Zeinalipour-Yazti, S. Lin, V. Kalogeraki, D. Gunopulos and W. Najjar, The 4th USENIX Conference on File and Storage Technologies (FAST'05), 2005.

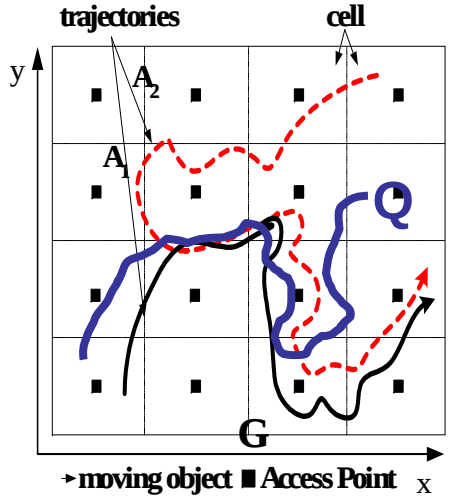
Related Activity 2: Retrieval using Score Bounds

- Suppose that each Node can only return **Lower** and **Upper Bounds** rather than **Exact scores**.
- e.g. instead of **16** it tells us that the similarity is in the range **[11..19]**

v1 id,lb,ub	**v2** id,lb,ub	**v3** id,lb,ub	**METADATA** id,lb,ub
A2,3,6	A4,4,5	A4,1,3	A4,10,18
A0,4,8	A2,5,6	A0,6,10	A2,13,19
A4,5,10	A0,5,7	A2,5,7	A0,15,25
A7,7,9	A3,5,6	A9,6,7	A3,20,27
A3,8,11	A9,8,10	A3,7,10	A9,22,26
A9,8,9	A7,12,13	A7,11,13	A7,30,35
....	....	....	....

m

n



- We developed two new algorithms: **UBK & UBLBK**
- Proposed in ***"Distributed Spatiotemporal Similarity Search",*** D. Zeinalipour-Yazti, S. Lin, D. Gunopulos, under review

References

TOP-K Query Processing & In-Situ Data Storage

– D. Zeinalipour-Yazti, Z. Vagena, D. Gunopulos, V. Kalogeraki, V. Tsotras, M. Vlachos, N. Koudas, D. Srivastava ***"The Threshold Join Algorithm for Top-k Queries in Distributed Sensor Networks",*** Proceedings of the 2nd international workshop on Data management for sensor networks DMSN (VLDB'2005), Trondheim, Norway, 2005.

– D. Zeinalipour-Yazti, S. Neema, D. Gunopulos, V. Kalogeraki and W. Najjar, ***"Data Acquision in Sensor Networks with Large Memories",*** IEEE Intl. Workshop on Networking Meets Databases NetDB

(ICDE'2005), Tokyo, Japan, 2005.

– D. Zeinalipour-Yazti, V. Kalogeraki, D. Gunopulos, A. Mitra, A. Banerjee and W. Najjar ***"Towards In-Situ Data Storage in Sensor Databases",*** 10th Panhellenic Conference on Informatics (**PCI'2005**) Volos, Greece, 2005.

Top-K Query Processing Techniques for Distributed Environments

by

Demetrios Zeinalipour

Thanks!

Wednesday, June 7th, 2006
"Mediteranean Studies" Seminar Room,
FORTH, Heraklion, Crete