Research introduction: XML partitioning for parallel holistic twig join and efficient ObjectRank estimation based on subgraphs

Toshiyuki Amagasa

Center for Computational Sciences

University of Tsukuba

XML Data Partitioning Strategies to Improve Parallelism in Parallel Holistic Twig Joins

Imam Machdi, Toshiyuki Amagasa, and Hiroyuki Kitagawa

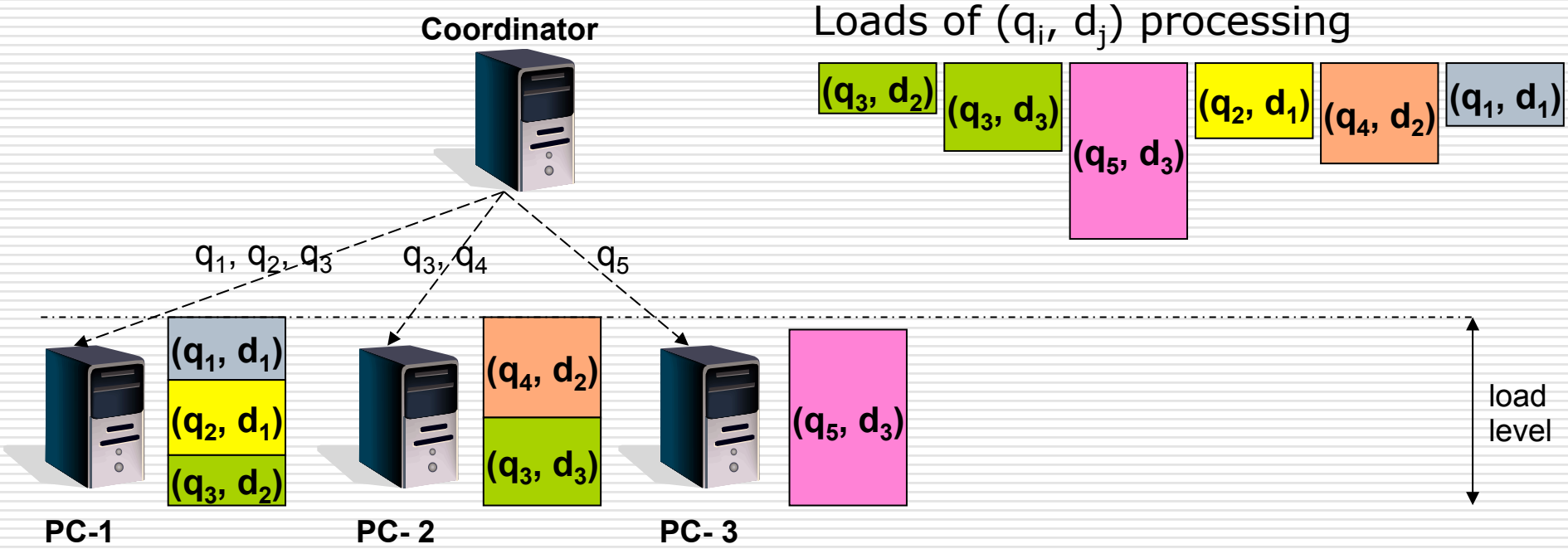
Graduate School of Systems and Information Engineering
Center for Computational Sciences
University of Tsukuba, Japan
machdi@kde.cs.tsukuba, {amagasa, kitagawa}@cs.tsukuba.ac.jp

Querying Large XML

- XML datasets keep growing
- XML query processing is computationally expensive
  - Given a tree pattern, find out all embedding in a huge XML document tree.
    - Holistic twig joins
- Commoditization of cluster computers

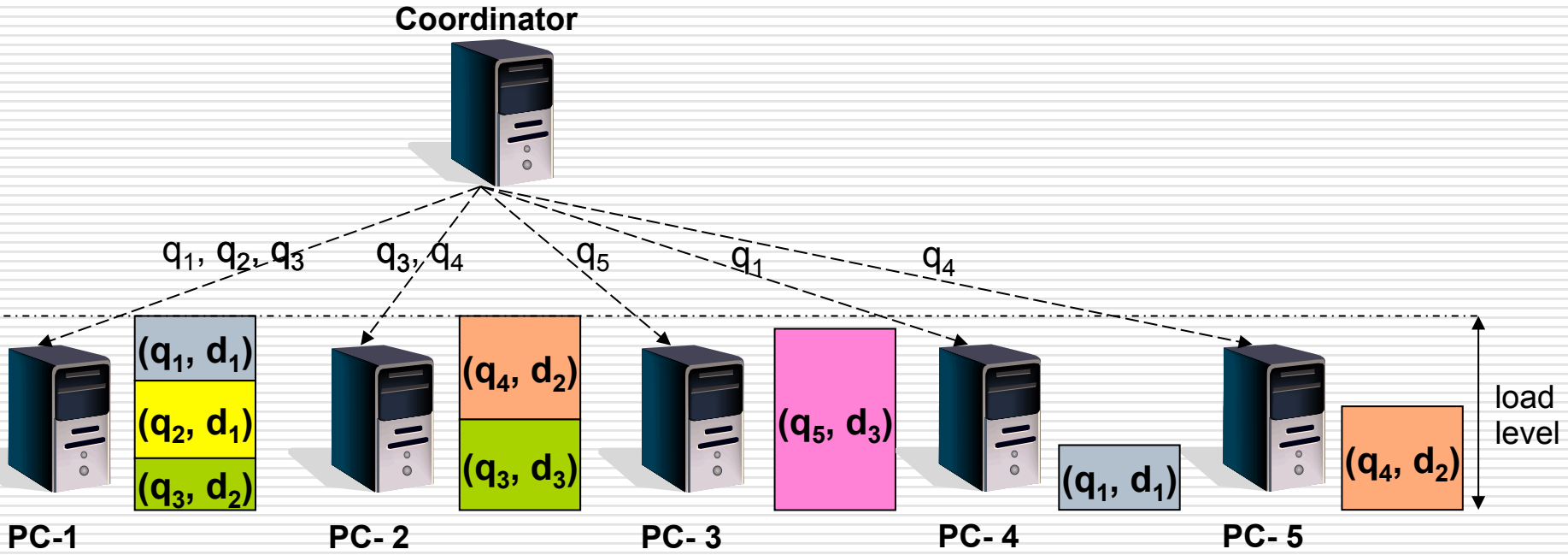
➔Using cluster computers to accelerate XML query processing

Query Processing on Cluster System



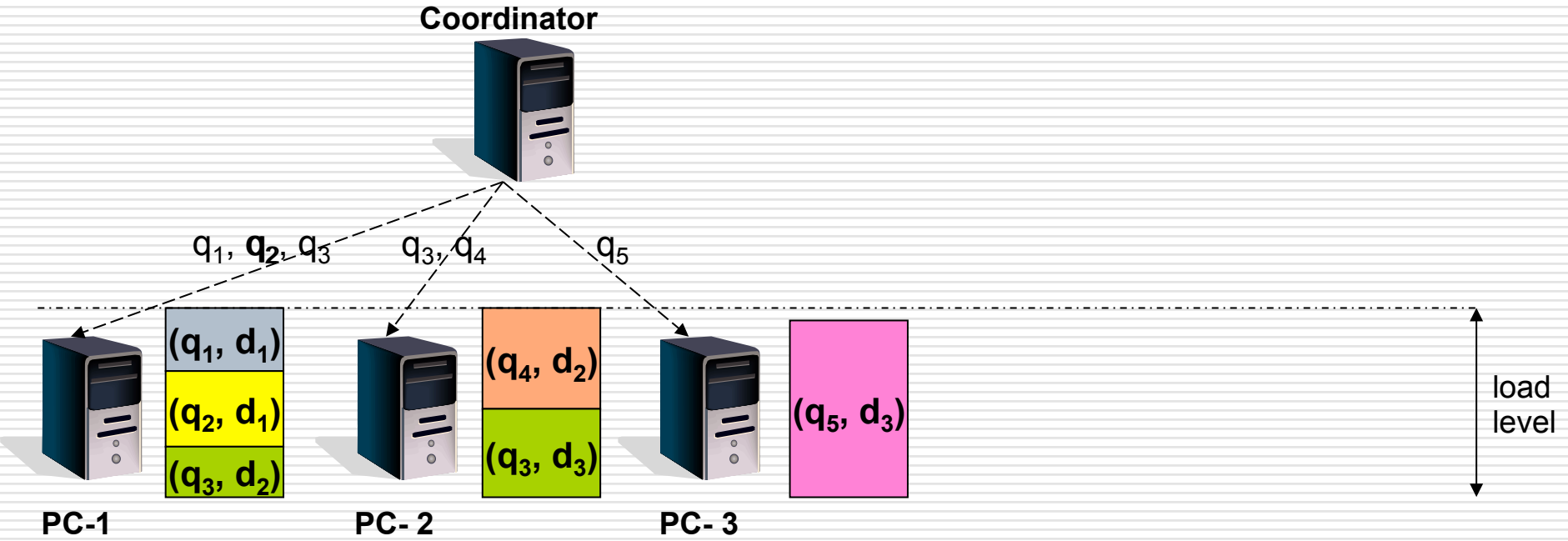
- Inter-query parallelism performs well due to balanced workloads.

1st Problem



- Adding more PCs leads to workload imbalance.

2nd Problem



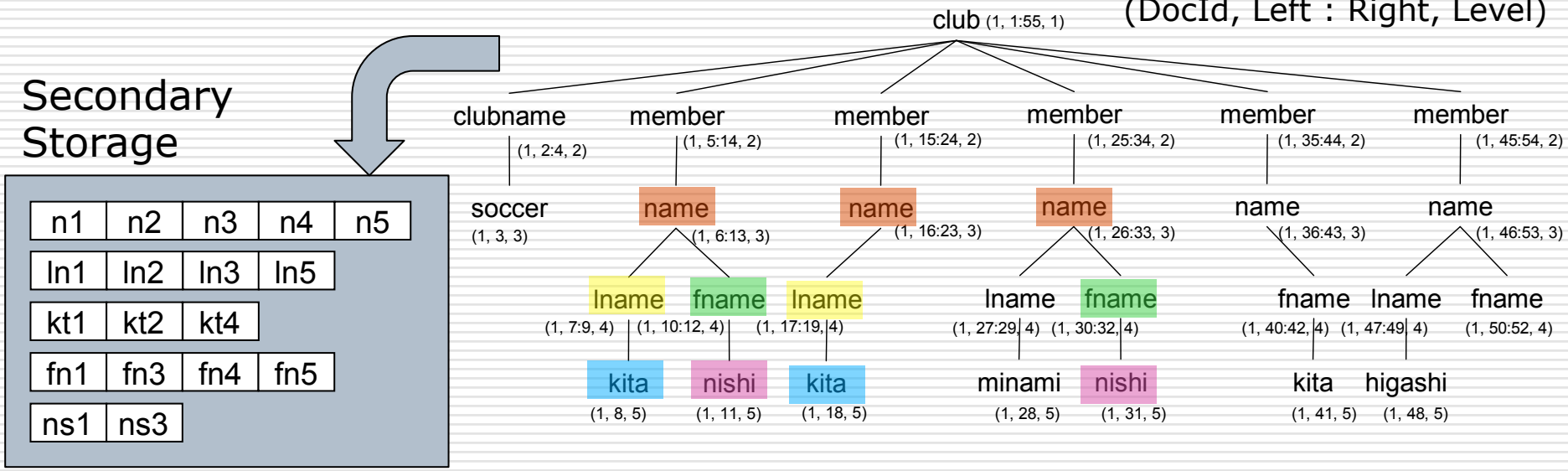
- Adding more PCs leads to workload imbalance.
- Executing few queries leads to low system utilization.

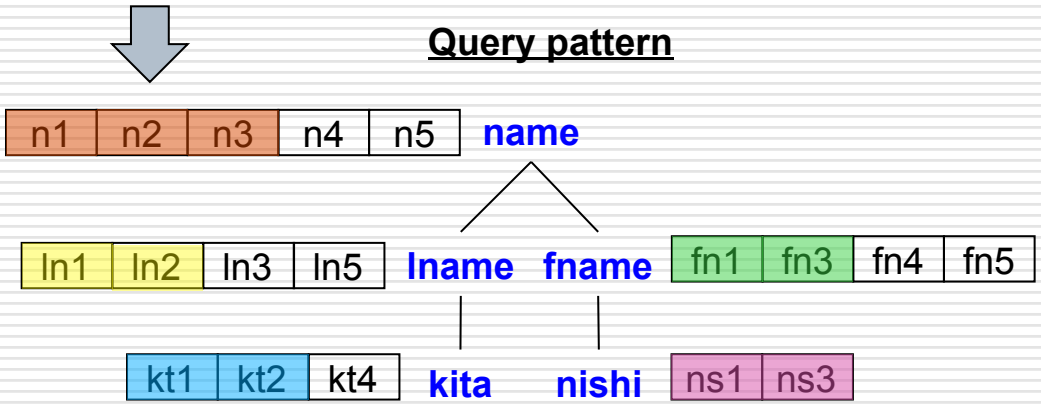
→ Both cause performance degradation.

Objective

- To provide XML data partitioning strategies for static and dynamic data distribution to achieve balanced workloads among cluster PCs.
- To improve performance of parallel query processing for both inter-query parallelism and intra-query parallelism.

Twig Stack Algorithm





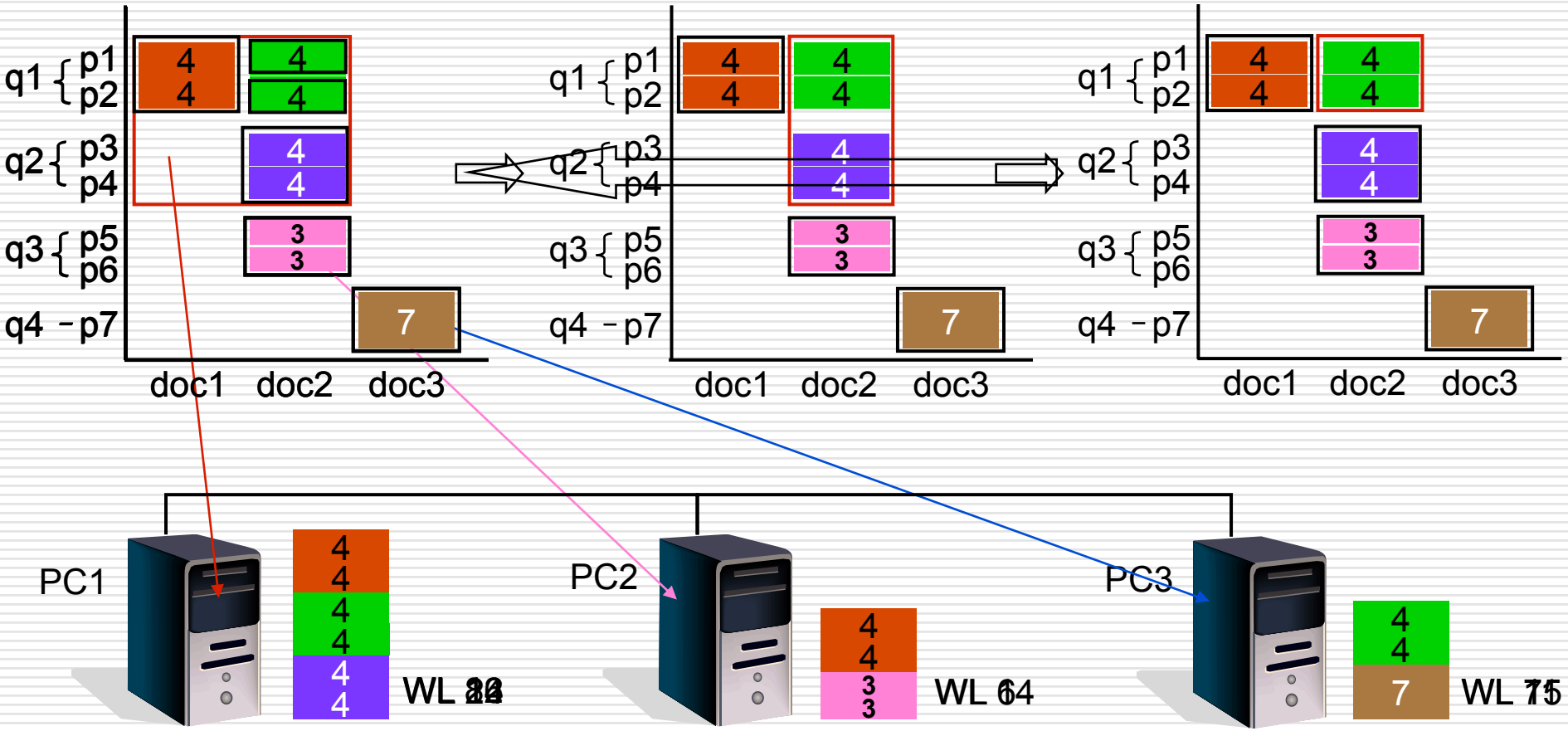
1st Phase: Partial Solutions

√ (n1) – (ln1) – (kt1)
√ (n1) – (fn1) – (ns1)
(n2) – (ln2) – (kt2)
(n3) – (fn3) – (ns3)

2nd Phase: Merge Partial Solutions

(n1) – (ln1) – (kt1) – (fn1) – (ns1)

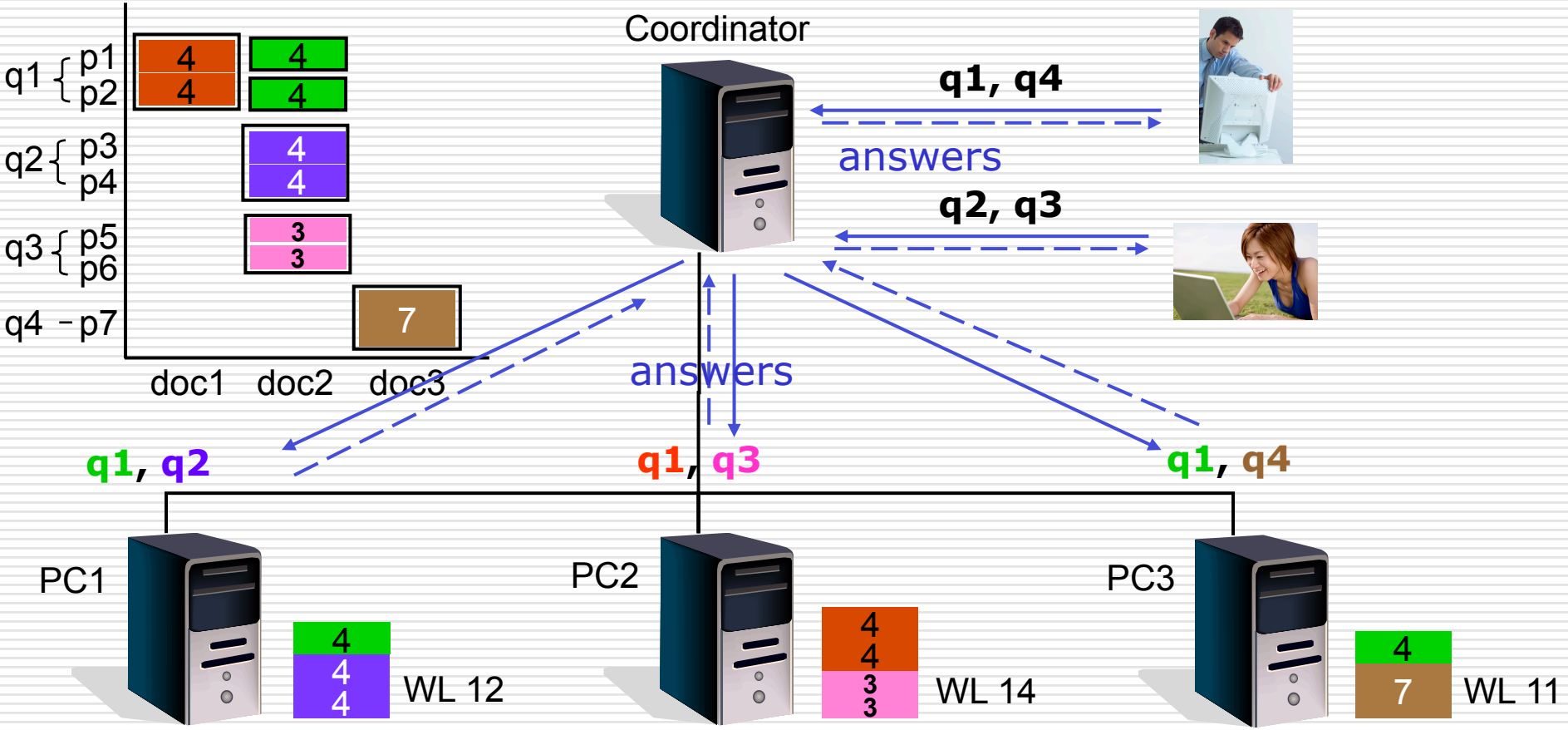
Static Data Distribution



Avg. WL = 12.33, Initial Variance = 102.33

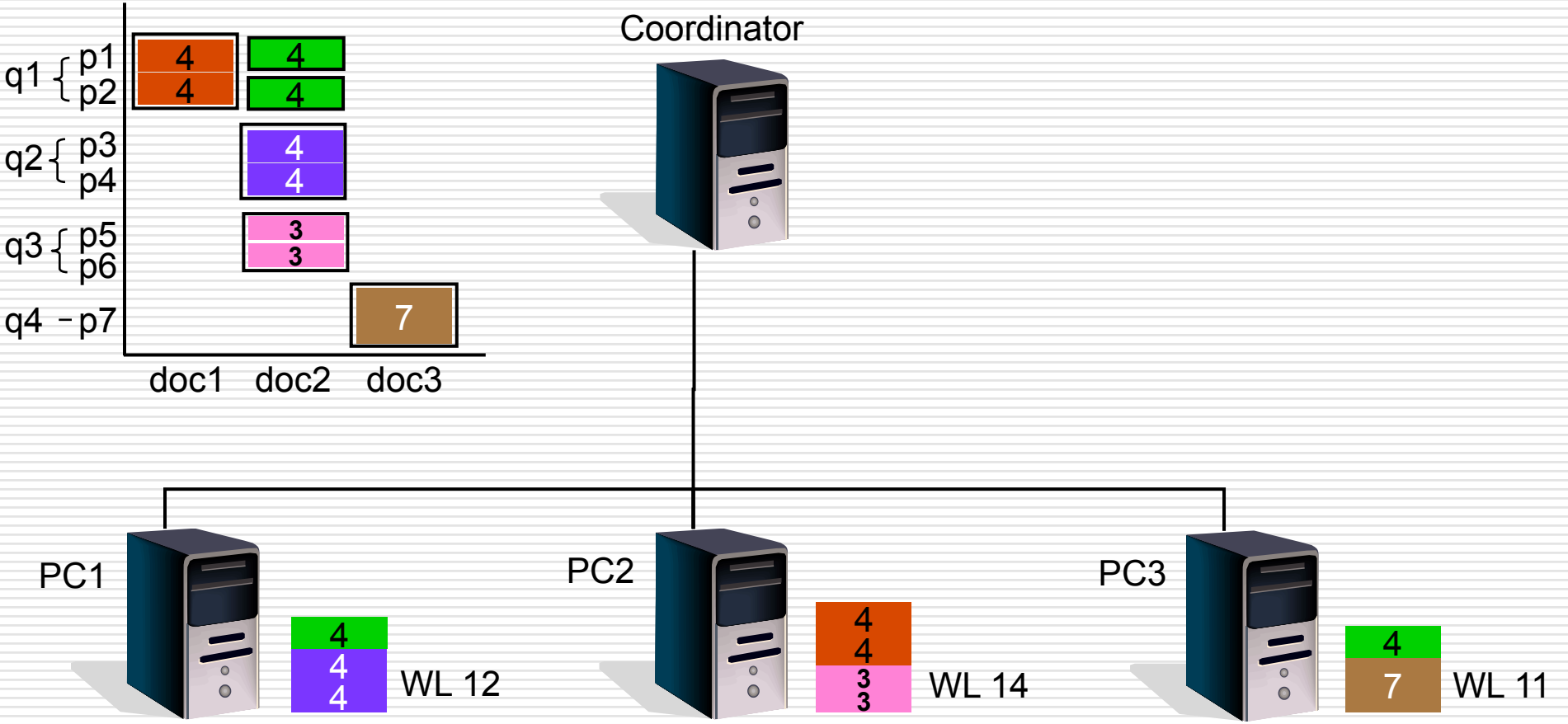
Current Variance = 22.33 (-98.0%)

Parallel Query Processing

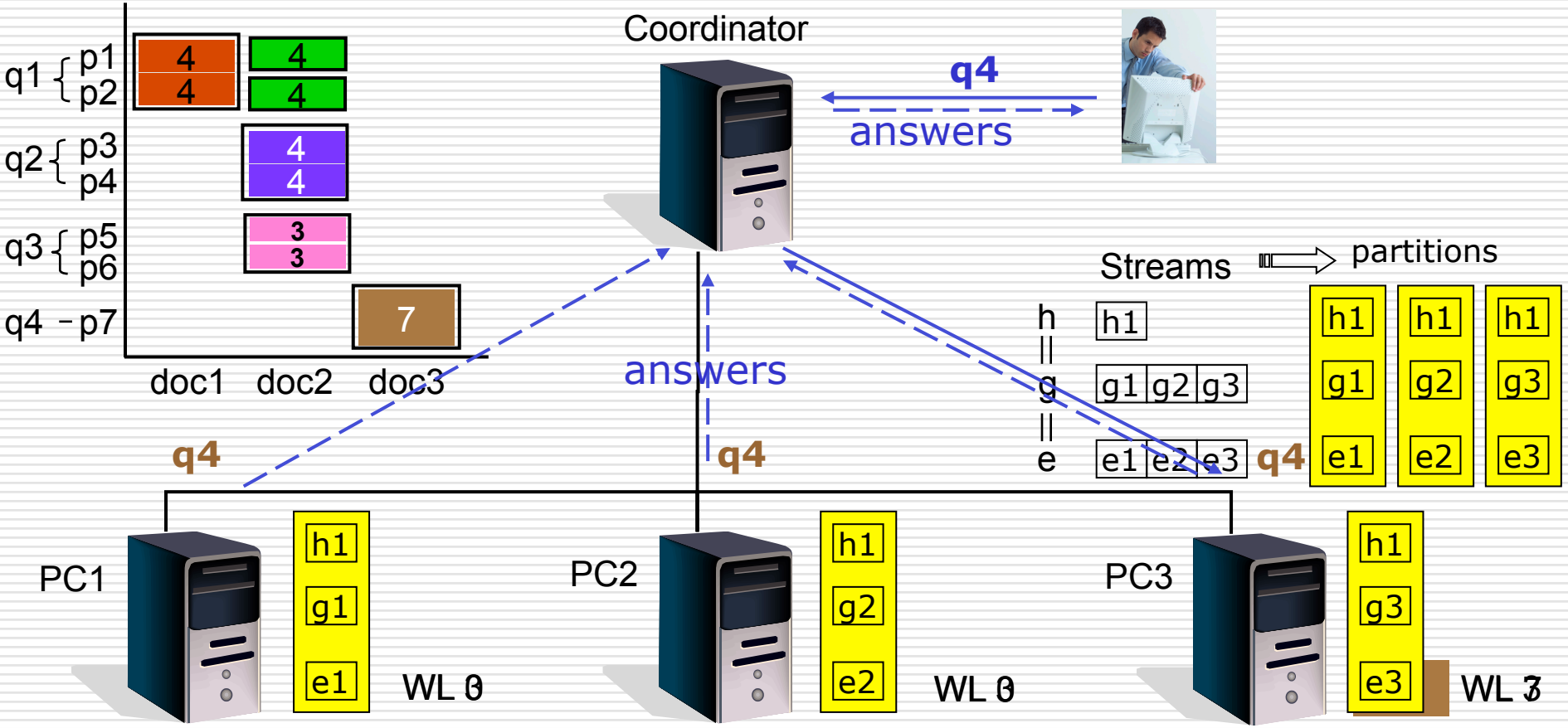


q1, q2, q3, q4 are executed for inter-query parallelism.
q1 is executed for intra-query parallelism.

Dynamic Data Distribution



Dynamic Data Distribution



Suppose only q4 is being executed in the system.
q4 is executed for intra-query parallelism.

XML Data Sets

No.	XML Data	DTD	# Queries	# Docs	Size (bytes)
1	Bibliography	1	4	16	158,096
2	Sport Clubs	1	3	12	162,986
3	Cars	1	6	48	1,357,856
4	Departments	1	5	19	2,722,723
5	Purchases	1	2	10	4,873,260
6	Quotes	1	2	10	4,412,418
7	Dramas	1	3	18	7,428,278
8	Sigmod 2002	3	10	43	2,934,193
9	Movies	5	21	5	28,463,633
10	Auction	1	4	8	119,900,420
11	Assembly	1	4	5	171,309,031
	Total	**17**	**64**	**194**	**343,722,894**
	Average				**1,771,767**
	Variance				**6.2E+13**

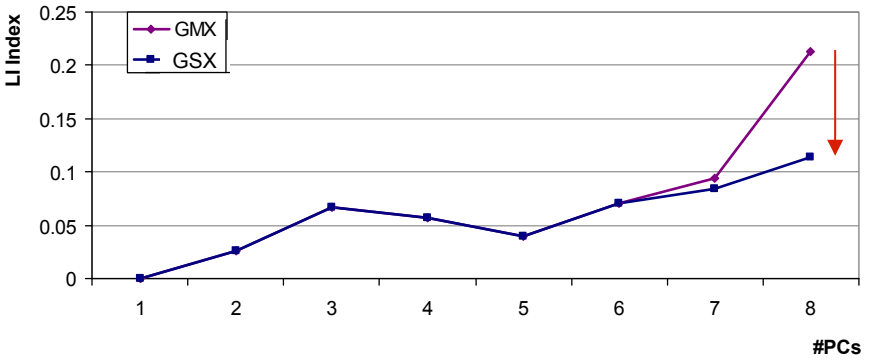
Static Data Distribution

Estimated Workload Distribution

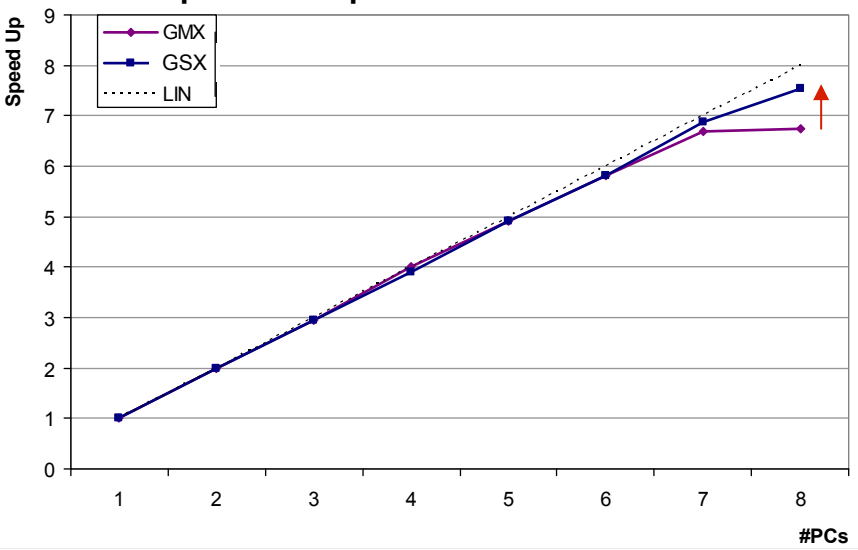
Cluster PC	Initial Workload	GMX Workload	GSX Workload
1	2,267.12	1,937.23	2,013.52
2	10,409.22	2,829.01	2,443.71
3	845.77	2,184.84	2,259.72
4	881.35	2,040.94	2,109.41
5	680.73	2,278.75	2,278.75
6	801.10	2,170.93	2,260.32
7	880.55	2,355.09	2,355.09
8	973.17	1,942.22	2,018.51
Total	17,739.01	17,739.01	17,739.01
Threshold	2,217.38	2,217.38	2,217.38
Variance	1.12E+07	8.36E+04	2.43E+04

- Workload estimation use a cost model.
- All 64 queries are executed simultaneously.
- LI is measured with actual execution time on each PC.
- Speed up is measured with the entire system execution time.

Load Imbalance Index



Speed Up Performance



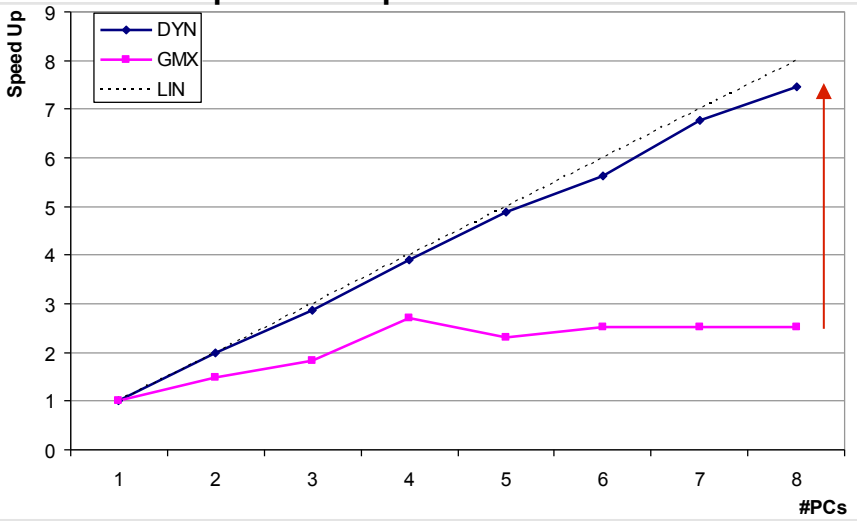
Dynamic Data Distribution

Selected queries

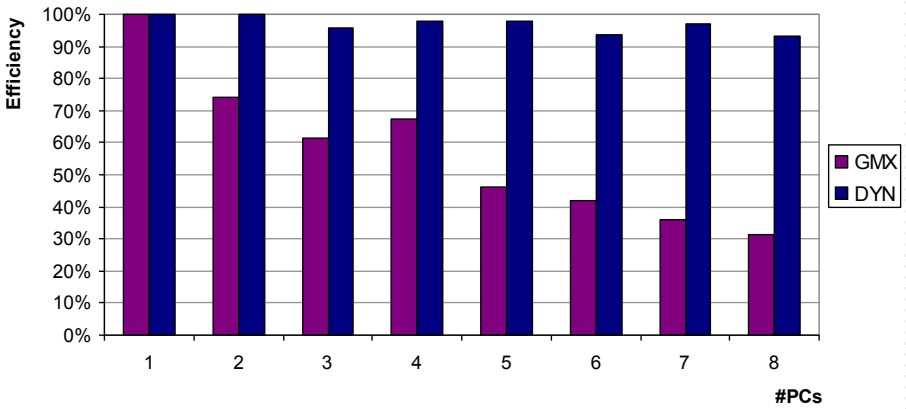
Query	XPath Expression	Stream Size
Q1	//article[/title/'Semantic'[/'Web']] [//author/'Wei'[/'Han']]//format/'PDF'	15,922
Q2	//company/Profile/EmployeeNumber [//state/WI]	40,000
Q3	//compositepart//atomicpart	1,110,510
Q4	//person[/name/'Kang'][/address[/city/'Panama']/Province]/Creditcard/6284	75,182
Q5	//m/t[/'Cleopatra']/a/'Arthur'	293,628

- 5 queries of 64 are selected randomly for simultaneous execution.
- System performance is improved.
- Efficiency of system utilization is better maintained.

Speed Up Performance



Efficiency



Conclusion and Future Work

- We proposed partitioning strategies for both static and dynamic XML data distribution.
- We provide different granularities:
  - Document cluster
  - Query cluster
  - Query
  - Sub-query
  - Stream of XML nodes

A Local Method for ObjectRank Estimation

Yuta Sakakura†, Yuto Yamaguchi†,
Toshiyuki Amagasa‡, Hiroyuki Kitagawa‡

† Graduate School of Systems and Information Engineering, University of Tsukuba

‡ Faculty of Engineering, Information and Systems, University of Tsukuba

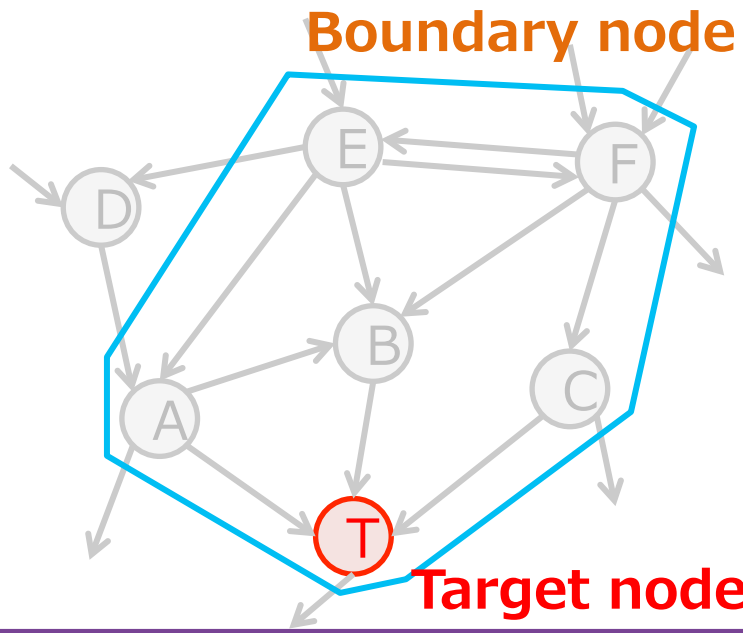
Graphs and Node Score Estimation

- Graphs are everywhere.
  - Social networks, biology, bibliography, etc.
- Estimating nodes' scores (importance) is one of the most popular and useful operations on graphs.
  - PageRank, HITS, etc.

- Demand for quick estimation of a specific node's score.
  - Relative comparison among few nodes
  - Temporal variation of a node's scores
  - Huge graph may be managed in a distributed databases

Chen's Method: Subgraph-based Method

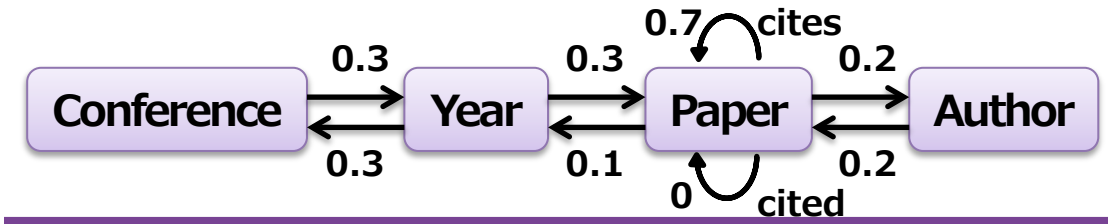
- The initial subgraph consists only of the target node.
- **Expand** the target node.
- Judge whether or not the newly added nodes have the **node scores** larger than or equal to the threshold.
- Expand the satisfied nodes recursively until there are no satisfied nodes.



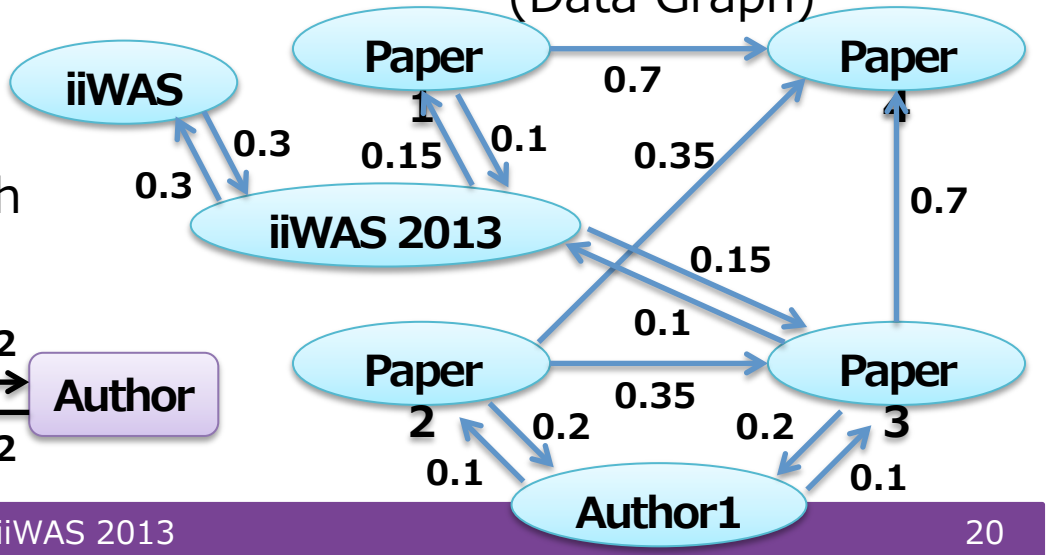
ObjectRank

- Method for evaluating the importance of nodes in a heterogeneous graph
- Link structure analysis method extended from PageRank
  - Deal with heterogeneous nodes and edges, distinguished by respective labels or weights.

Authority Transfer Schema Graph
(Schema Graph)



Authority Transfer Data Graph
(Data Graph)



Objective and Approach

- Develop a new method to quickly estimate the ObjectRank score in edge-weighted graphs.
  - We need to take into account different kinds of nodes as well as edge weights.

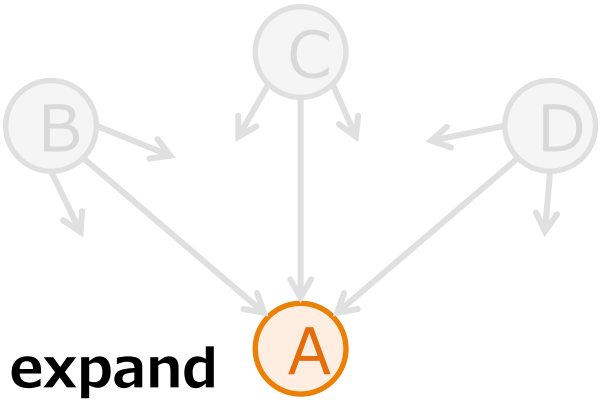
- Basic approach
  - Based on Chen's subgraph-based method.
  - Modify the algorithm by taking into account edge weights.

Basic Idea

- Both methods construct a local graph by expanding boundary nodes recursively.
- When expanding, different operation is conducted.

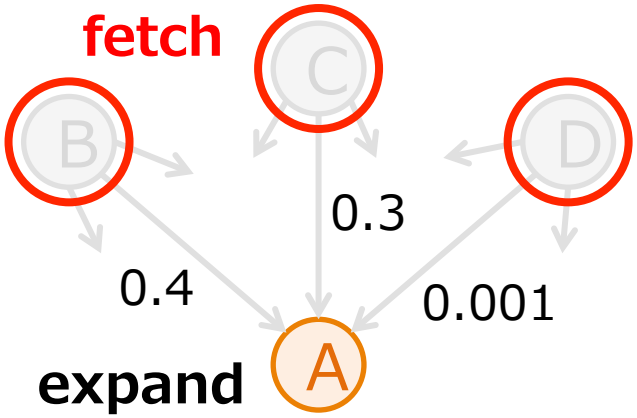
Chen's Method

Add all edges



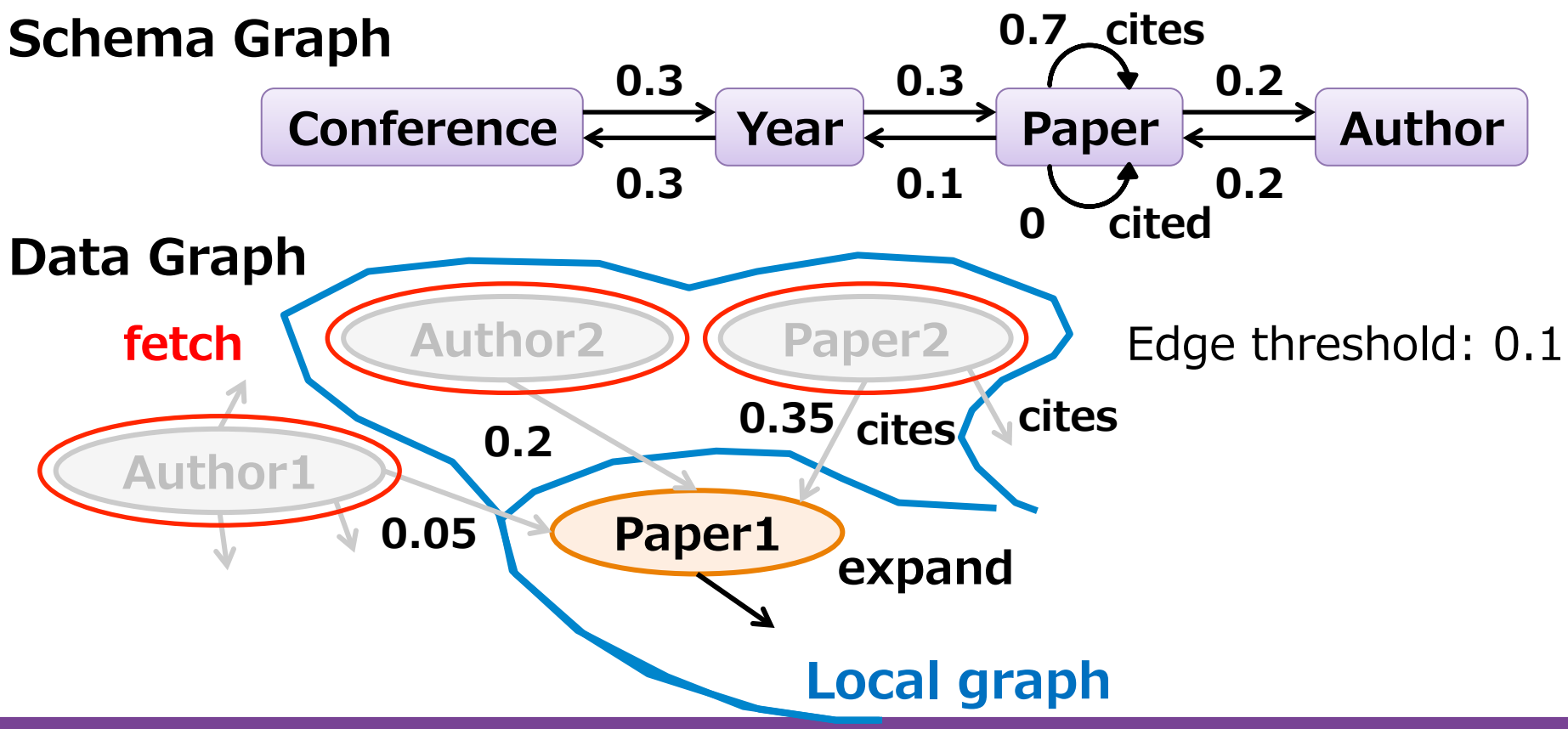
Proposed Method

Prune some edges



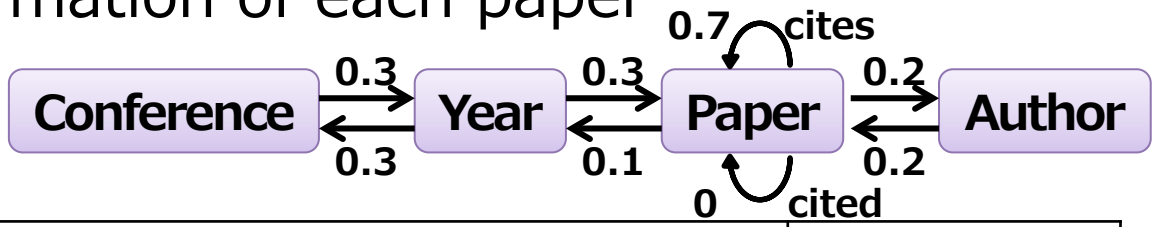
Proposed Method

- Prune edges with small weights in Data Graph
  - Set the threshold of edge weights (edge threshold)



Dataset

- DBLP dataset offered by Arnetminer
  - The following information of each paper
    - Title
    - Authors
    - Publisher
    - Published year
    - Citation



# of Paper nodes	1,553,079
# of Author nodes	915,096
# of Conference nodes	6,246
# of Year nodes	28,669
# of the edges (Conference – Year)	57,338
# of the edges (Year – Paper)	3,106,158
# of the cites/cited edges	2,052,921
# of the edges (Paper – Author)	8,219,584

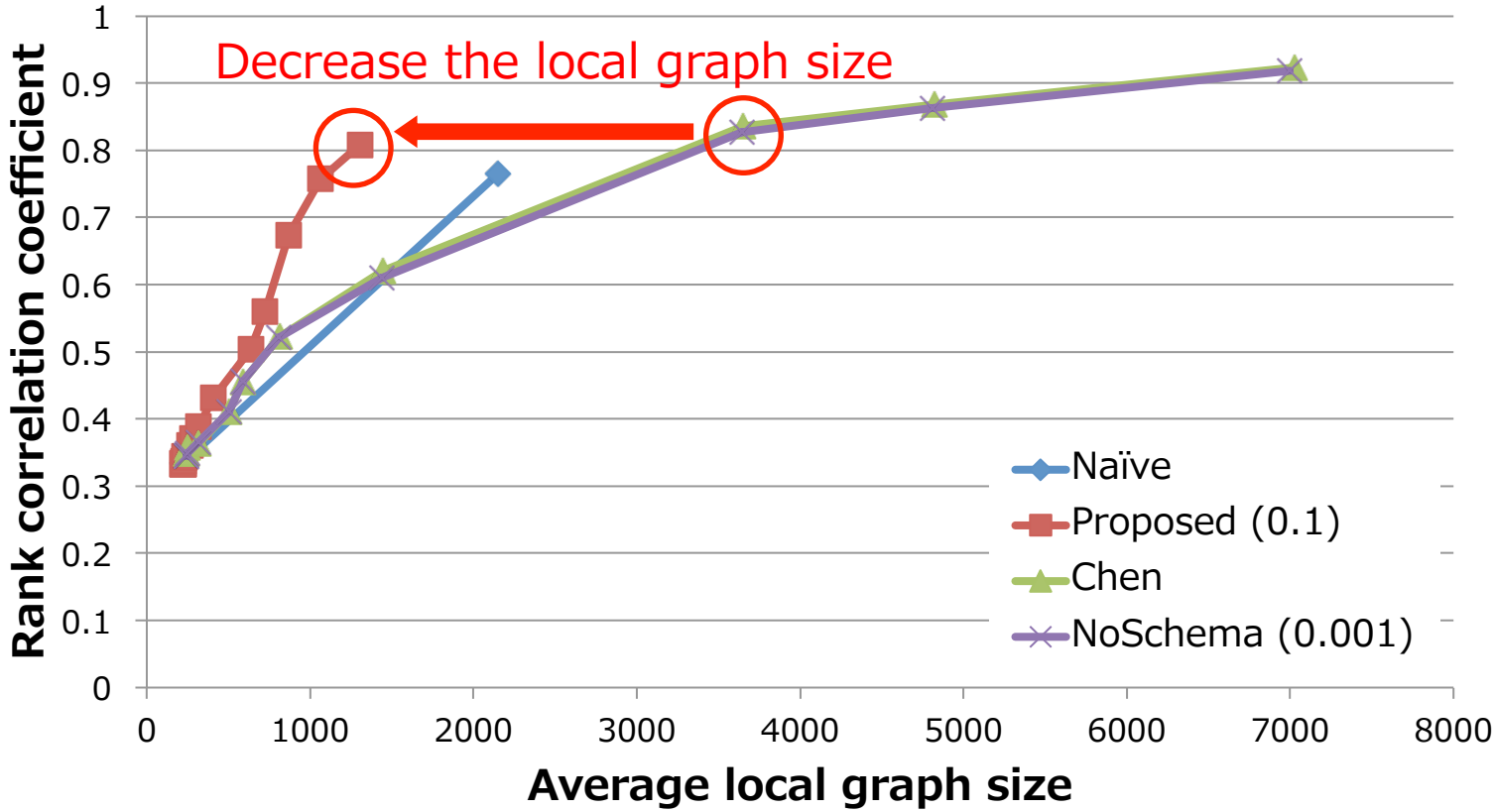
Comparative Methods

- *Proposed* : Our proposed method
- *Chen* : Chen's method
- *NoSchema*
  - The proposed method without considering Schema Graph
- *Naïve*
  - Construct a local graph including all nodes whose shortest path to the target node is within k

	Prune edges	Consider Schema Graph
Proposed	○	○
Chen	×	×
NoSchema	○	×

Results
Accuracy VS Local Graph Size

- Set the edge threshold which achieves the best performance for *Proposed* and *NoSchema.*
- Results varying the node threshold or k



Conclusion and Future Work

- Conclusion
  - Local method for ObjectRank estimation
  - Experiments show that the proposed method decreases the local graph size while maintaining the estimation accuracy.
- Future Work
  - Evaluate the proposed method using different dataset or Schema Graph.
  - Automatic way to decide the appropriate thresholds

— Introduction to Current Research Work—

Continuous Outlier Detection on Uncertain Data Streams

Salman Ahmed SHAIKH
(PhD Student)

advised by
Hiroyuki Kitagawa

Department of Computer Science
University of Tsukuba, Japan

February 19, 2014

筑波大学
University of Tsukuba

Outline

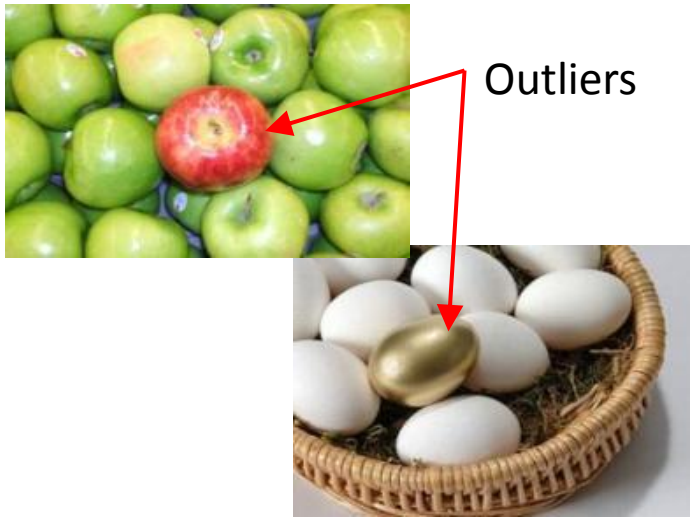
- Introduction and Background
- Problem Definition
- Outlier Detection on Uncertain Static Data (UDB)
- Continuous Outlier Detection on Uncertain Data Streams (CUDB)
- Experimental Evaluation
- Conclusion and Future Work

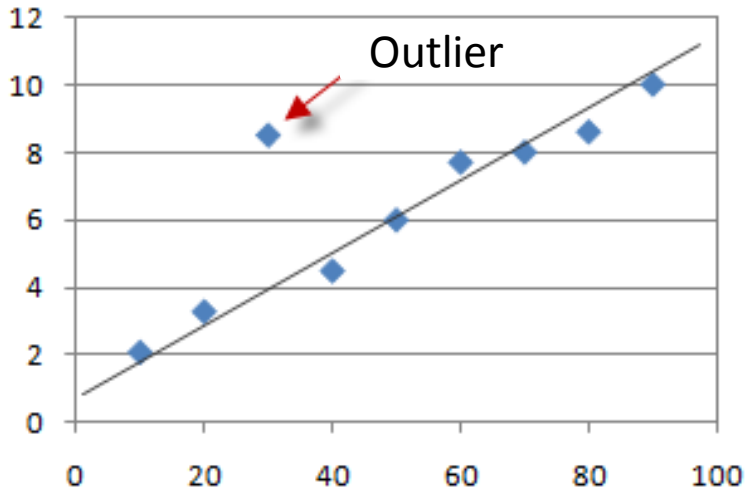
What is an Outlier?

- A data point which is significantly different from the remaining data.

Definition (Outlier)

An observation which deviates so much from the other observations as to arouse suspicion that it was generated by a different mechanism [Hawkins 1980].





Applications of Outlier Detection

- Outlier detection is a key problem in data mining and it has several applications, such as:
  - Network intrusion detection [Chandola et al. 2007]
  - Credit card fraud detection [Bolton et al. 1999]
  - Malfunctioning sensors identification [Zhang et al. 2007]
  - Industrial damage detection [Hollier et al. 2002]
  - Medical anomaly diagnosis [Agarwal et al. 2005]
  - Textual anomaly detection [Srivastava 2006]
  - etc.

Outlier Detection Approaches

- In data mining, several outlier detection approaches have been proposed, such as:
  - Distance based [Knorr et al. 1998], [Maria et al. 2011]
  - Clustering based [Jiang et al. 2000], [Budalakoti et al. 2006]
  - Density based [Breunig et al. 2000], [Lian et al. 2012]
  - Classification based [Hawkins 2002] [Tandon et al. 2007]
- Why distance-based approach?
  - The simplest and the most basic one.
  - Can be used as preprocessing before applying more sophisticated outlier detection techniques.

Distance-based Outlier Detection on Deterministic Data by Knorr et al.†

Definition (DB-Outlier: Distance-based Outlier on Deterministic Data)

An object *o* is a *DB(p,D)* outlier, if at least fraction *p* of the objects are outside distance *D* of *o*.

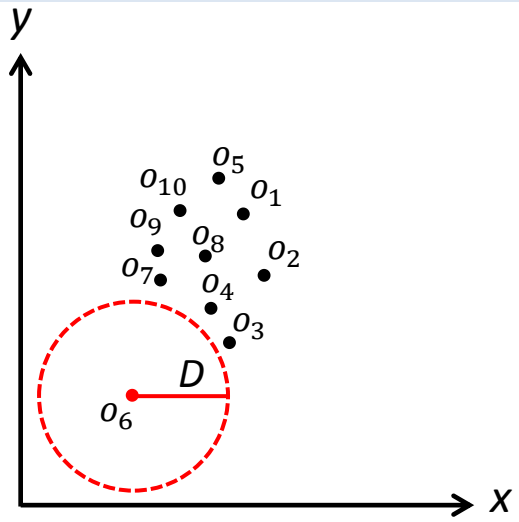


Fig.: Example of *DB(p,D)* outlier

Dataset: $\{o_1, o_2, \ldots, o_{10}\}$
p: (specified by user)
D: (specified by user)

Example (DB-Outlier)

Input: $\{o_1, o_2, \ldots, o_{10}\}$, *D*, *p*

Output: o_6

† E.M. Knorr and R.T. Ng.: Algorithms for mining distance-based outliers in large datasets. In VLDB, 1998.

Outline

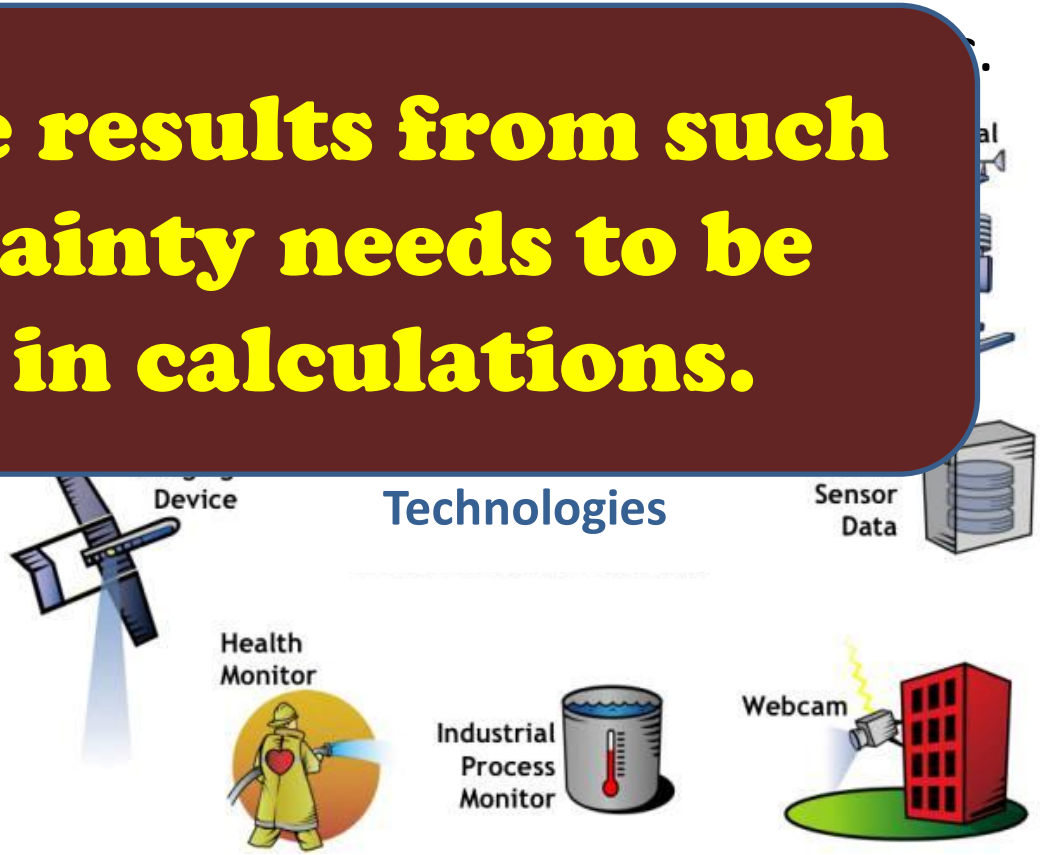
- Introduction and Background
- Problem Definition
- Outlier Detection on Uncertain Static Data (UDB)
- Continuous Outlier Detection on Uncertain Data Streams (CUDB)
- Experimental Evaluation
- Conclusion and Future Work

Data Uncertainty

- Recently, advancement in data collection technologies have resulted in a vast amount of uncertain static and stream data.

To get reliable results from such data, uncertainty needs to be considered in calculations.

- Delay/loss of data in transfer
- Privacy concerns
- etc.



Distance-based Outlier on Uncertain Static Data (UDB-Outlier)†

Definition (UDB-Outlier: Uncertain Distance-based Outlier)

An uncertain object *o* is a distance-based outlier if the expected number of objects lying outside the distance *D* of *o* are at least fraction *p*.

Proposed Solution

Continuous Outlier Detection on Uncertain Data Streams.

Fig.: Example of UDB-outlier

Input: $\{o_1, o_2, \ldots, o_{10}\}$, p, D, σ

Output: o_6

† S.A. Shaikh and H. Kitagawa,: "Efficient Distance-based Outlier Detection on Uncertain Datasets of Gaussian Distribution", WWWJ, 2013.

Continuous Outlier Detection (CUDB Outlier Detection)

- Moving objects change their states (i.e., locations) over time.
- These states arrive as time series data streams.

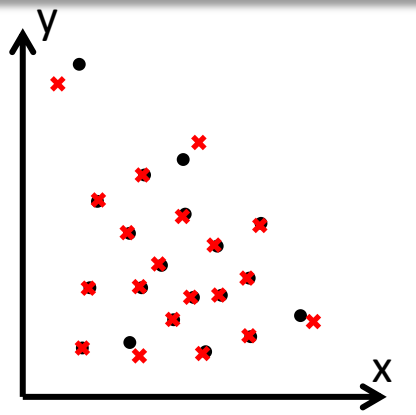


Table: Arrival of time series data streams.

Time	State set	o_1	o_2	...	o_N
t_1	S^1	$\overrightarrow{\mathcal{A}_1^1}$	$\overrightarrow{\mathcal{A}_2^1}$	...	$\overrightarrow{\mathcal{A}_N^1}$
t_2	S^2	$\overrightarrow{\mathcal{A}_1^2}$	$\overrightarrow{\mathcal{A}_2^2}$	...	$\overrightarrow{\mathcal{A}_N^2}$
⋮	⋮	⋮	⋮	⋮	⋮
t_M	S^M	$\overrightarrow{\mathcal{A}_1^M}$	$\overrightarrow{\mathcal{A}_2^M}$	...	$\overrightarrow{\mathcal{A}_N^M}$

$\overrightarrow{\mathcal{A}_i^j}$ is an attribute vector, denoting the state of an object o_i at time t_j.

Definition (CUDB-Outlier: Continuous UDB Outlier)

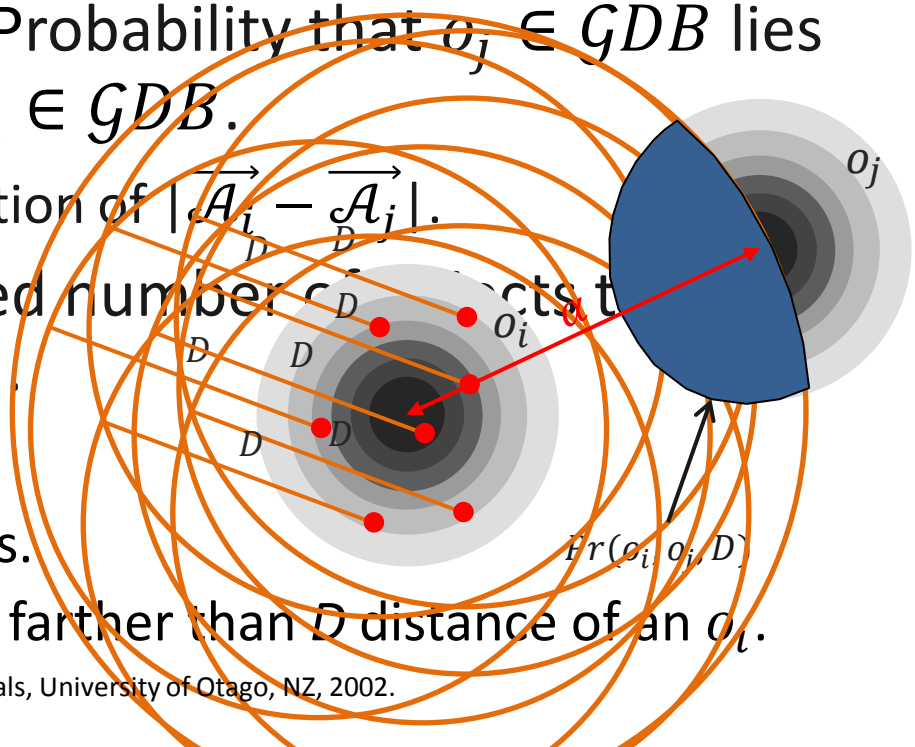
An uncertain object *o* is an outlier in a state set S^j, if its state differs greatly from other objects' states at time t_j.

Outline

- Introduction and Background
- Problem Definition
- Outlier Detection on Uncertain Static Data (UDB)
- Continuous Outlier Detection on Uncertain Data Streams (CUDB)
- Experimental Evaluation
- Conclusion and Future Work

Preliminaries (1/2)

- $\mathcal{GDB} = \{o_1, \ldots, o_N\}$: Set of uncertain objects.
  - **Assumptions**:
    - Attributes ($\overrightarrow{\mathcal{A}_i}$) of o_i are uncertain and this uncertainty is given by the Gaussian distribution.
    - $\overrightarrow{\mathcal{A}_i}$ are uncorrelated and its std. deviations are uniform in all dimensions. *(A statistical procedure Principal Component Analysis† can be used to transform a correlated Gaussian distribution into an uncorrelated one)*
- $\Pr(o_i, o_j, D)$ or $\Pr(\alpha, D)$: Probability that $o_j \in \mathcal{GDB}$ lies within the *D* distance of $o_i \in \mathcal{GDB}$.
  - Computed from the distribution of $|\overrightarrow{\mathcal{A}_i} - \overrightarrow{\mathcal{A}_j}|$.
- $\#D\text{-neighbors}(o_i)$: Expected number of objects t… within the *D* distance of o_i.
- $\theta = N(1-p)$: Threshold.
  - N: number of dataset objects.
  - *p*: fraction of objects that lie farther than *D* distance of an o_i.



† Lindsay I Smith. *A tutorial on Principal Components Analysis*. Student Tutorials, University of Otago, NZ, 2002.

Preliminaries (2/2)

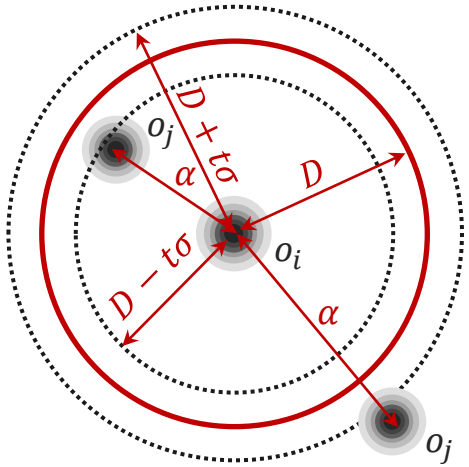
Lemma: Approximate $Pr(o_i, o_j, D)$ Values

- If attributes $\overrightarrow{\mathcal{A}_i}$, $\overrightarrow{\mathcal{A}_j}$ of o_i, o_j follow the Gaussian distribution, then $|\overrightarrow{\mathcal{A}_i} - \overrightarrow{\mathcal{A}_j}|$ also follows the Gaussian distribution $N(\overrightarrow{\mu_i} - \overrightarrow{\mu_j}, \Sigma_\mathrm{i} + \Sigma_\mathrm{j})^\dagger$.

Let α denotes the ordinary Euclidean distance between the means of objects o_i, o_j.

- If $\alpha \leq D - t\sigma$, then $Pr(o_i, o_j, D) \approx 1$.
- If $\alpha \geq D + t\sigma$, then $Pr(o_i, o_j, D) \approx 0$.

where t is large enough to enclose a large portion $(say > 99\%)$ of the Gaussian distribution.



†Weisstein, Eric W. "Normal Difference Distribution." From *MathWorld*--A Wolfram Web Resource.

Naïve Approach

- Nested loop is used to find the $\#D$-neighbors of each $o_i \in \mathcal{G}DB$.
- On average, $O(N^2)$ costly probability computations are required (e.g., 2D computation).

$$\Pr(o_i, o_j, D) = \int_R (\overrightarrow{\mathcal{A}_i} - \overrightarrow{\mathcal{A}_j})\, \mathrm{d}\vec{\mathcal{A}} = \frac{1}{4\pi\sigma^2} \int_0^D \int_0^{2\pi} \exp\left(\frac{-1}{4\sigma^2}(r^2 - 2\alpha r cos\theta + \alpha^2)\right) r d\theta dr\,.$$

- Naïve approach is too costly.

Proposed solution: Cell-based Outlier Detection

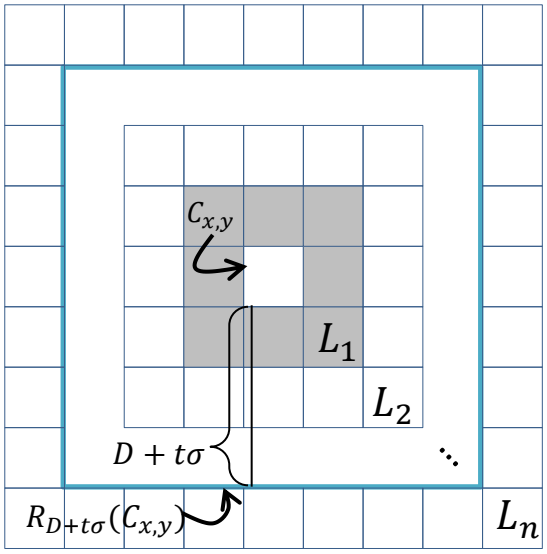
Objective: *To reduce the number of costly probability computations.*

Cell-based Outlier Detection

- Aimed at reducing the number of costly probability computations.
- Cell-Grid Structure (2-dimensional)
  - Let $\mathcal{GDB}$ contains 2-dimensional data objects.
  - Each object's mean in $\mathcal{GDB}$ is mapped to a 2D space, which is partitioned into cells of length l.
  - $C_{x,y}$ is a cell in the grid $\mathcal{G}$.
    - $L_1, \ldots, L_n$ denotes the neighboring layers of cell $C_{x,y}$.
    - L_1 cells of $C_{x,y}$ are given by

$$L_1(C_{x,y}) = \{C_{u,v} | u = x \pm 1, v = y \pm 1, C_{u,v} \neq C_{x,y}\}$$

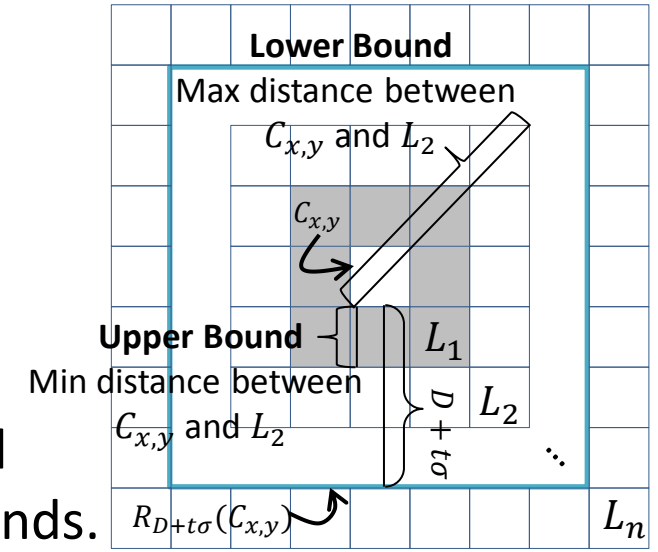
    - $L_2, \ldots, L_n$ are defined in a similar way.



  - Region $R_{D+t\sigma}(C_{x,y})$: For each $o_i \in C_{x,y}$ and $o_j \notin R_{D+t\sigma}(C_{x,y})$, $Pr(o_i, o_j, D) \approx 0$.

Cell-based Pruning

- **Cell Pruning:** Cell bounds on $\#D$-neighbors are computed for each grid cell to prune them.
  - Bounds Computation
    - Distance between cell and its layers, their object counts, and pre-computed $\Pr(\alpha, D)$ are used to compute cell bounds.
    - Layer by layer contribution of $\#D$-neighbors, for bounds computation of a cell $C_{x,y}$, is made only by layers within region $R_{D+t\sigma}(C_{x,y})$.
    - All the objects outside region $R_{D+t\sigma}(C_{x,y})$, contribute for bounds computation as a single layer.
  - Pruning
    - If $LB(C_{x,y}) > \theta$, all the objects in $C_{x,y}$ are inliers.
    - If $UB(C_{x,y}) \leq \theta$, all the objects in $C_{x,y}$ are outliers.



Outline

- Introduction and Background
- Problem Definition
- Outlier Detection on Uncertain Static Data (UDB)
- Continuous Outlier Detection on Uncertain Data Streams (CUDB)
- Experimental Evaluation
- Conclusion and Future Work

Continuous Outlier Detection (CUDB Outlier Detection)

○ Simple Approach:

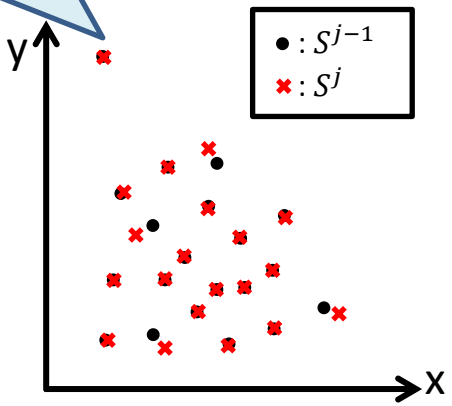
- Perform UDB outlier detection for every state set S^j.

State of all the objects may not change much between two time stamps.

Table: Arrival of time series data streams.†

Time	State set	o_1	o_2	...	o_N
t_1	S^1	$\overrightarrow{\mathcal{A}_1^1}$	$\overrightarrow{\mathcal{A}_2^1}$	...	$\overrightarrow{\mathcal{A}_N^1}$
t_2	S^2	$\overrightarrow{\mathcal{A}_1^2}$	$\overrightarrow{\mathcal{A}_2^2}$	...	$\overrightarrow{\mathcal{A}_N^2}$
⋮	⋮	⋮	⋮	⋮	⋮
t_{M-1}	S^{M-1}	$\overrightarrow{\mathcal{A}_1^{M-1}}$	$\overrightarrow{\mathcal{A}_2^{M-1}}$	...	$\overrightarrow{\mathcal{A}_N^{M-1}}$
t_M	S^M	$\overrightarrow{\mathcal{A}_1^M}$	$\overrightarrow{\mathcal{A}_2^M}$	...	$\overrightarrow{\mathcal{A}_N^M}$

†Yellow color $\overrightarrow{\mathcal{A}_i^j}$ shows the state change objects.



State sets S^{j-1} & S^j

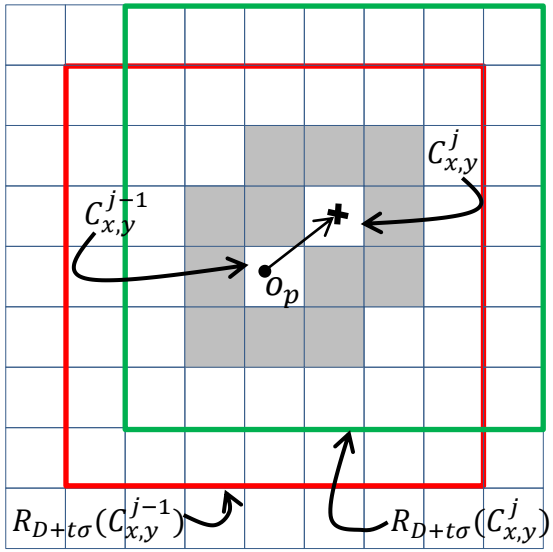
Proposed solution:

Incremental outlier detection based on the change between state sets S^{j-1} and S^j

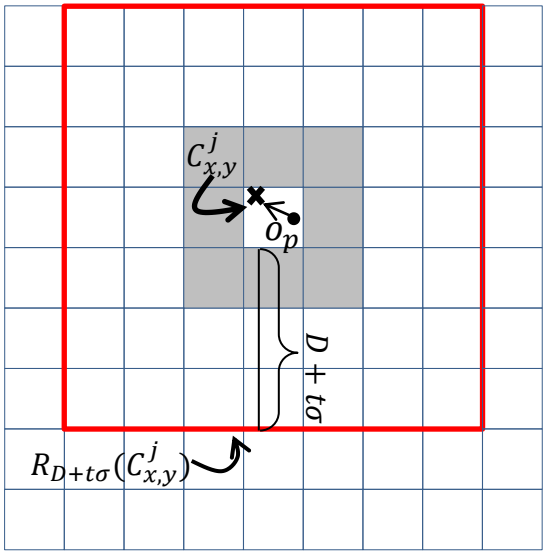
Incremental Outlier Detection: Key Idea (1/2)

- Process only state-change objects at current time t_j.
  - State-Change Objects (SC-Objects):
    - Objects whose states have changed from time t_{j-1} to t_j.
    - Let $C_{x,y}^{j}$ denotes a cell $C_{x,y}$ at time t_j; within the cell-grid† o_p can change state in the following ways.

Case 1: Move to a different cell



Case 2: Move within a cell



Incremental Outlier Detection: Key Idea (2/2)

- Arrival of objects' new states:
  - Re-outlier detection is required only for the cells affected by the SC-Objects (Target cells).
- Target cells are of the following 3 types.

Type A: Cells containing the SC-Objects which have moved to or from another cell at time t_j (Case 1).

Type B: Cells in regions $R_{D+t\sigma}$ of Type A cells.

Type C: Un-pruned cells of the grid $\mathcal{G}$.

Main cost of the Continuous outlier detection algorithm lies in the processing of Type C cells, due to the expensive #D-neighbors computation.

Incremental Processing of Un-pruned Objects

- Un-pruned objects processing require costly $\#D$-neighbors computation.
- This cost can be reduced by utilizing the $\#D$-neighbors computed in the previous state.
  - A Hash table is used to store $\Pr(o_p, o_q, D)$ values computed at time t_{j-1}.
  - At time t_j, these values can be retrieved in $O(1)$ time.
  - $\Pr(o_p, o_q, D)$ values are computed in two cases at time t_j.
    1. States of o_p, o_q or both have changed.
    2. $\Pr(o_p, o_q, D)$ is not available in the Hash table.

Outline

- Introduction and Background
- Problem Definition
- Outlier Detection on Uncertain Static Data (UDB)
- Continuous Outlier Detection on Uncertain Data Streams (CUDB)
- Experimental Evaluation
- Conclusion and Future Work

Experimental Setup

Datasets

Dataset	Type	Description	Dim.	Size
TG	Synthetic	Tri-modal Gaussian	2	5,000
MOW†	Real	3 hourly weather forecast data	2	5,802

SC-Objects

- **TG**: Generated by adding normal random numbers with zero mean and std. deviation = σ_{SC}= 5 to the fraction of the dataset values.
- **MOW**: Consecutive forecasts are used as data streams.

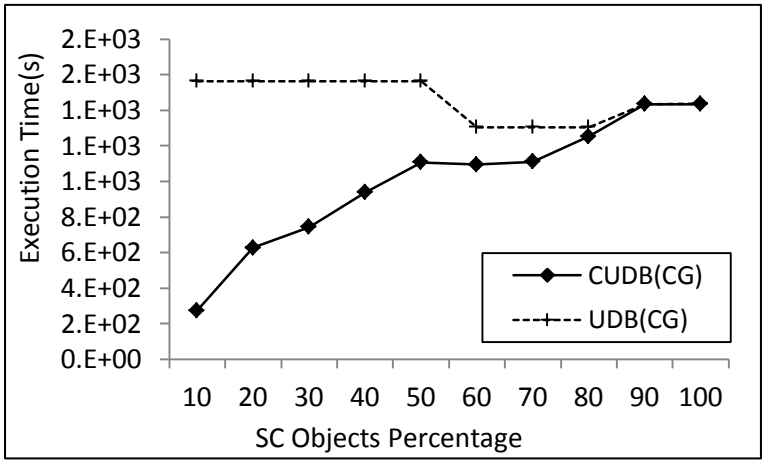
Compared Methods

- **CUDB(CG)**: CUDB outlier detection using conventional Gaussian uncertainty.
- **UDB(CG)**: UDB outlier detection using conventional Gaussian uncertainty, executed for every timestamp.

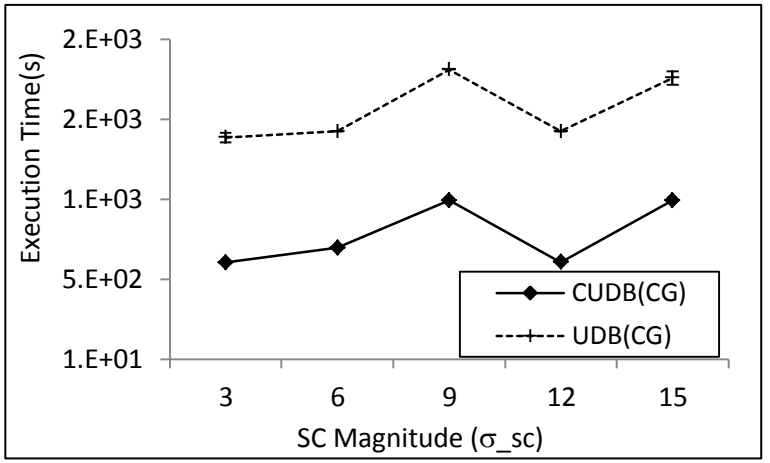
† Metoffice weather forecast data: http://www.metoffice.gov.uk/

Efficiency Evaluation (1/2)

- Varying SC-Objects and SC-Magnitude
  - Target cells increase with the increase in SC-Objects' percentage and SC-Magnitude.



Varying SC-Objects ($\sigma_{SC} = 5$)

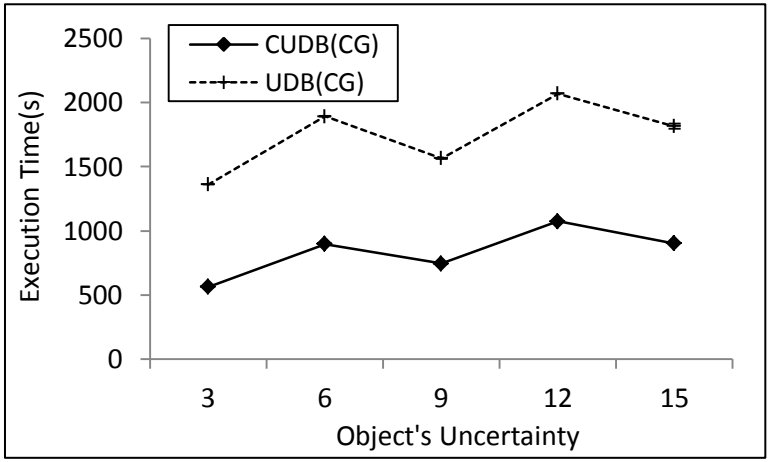


Varying SC Magnitude (SC-Objects = 30%)

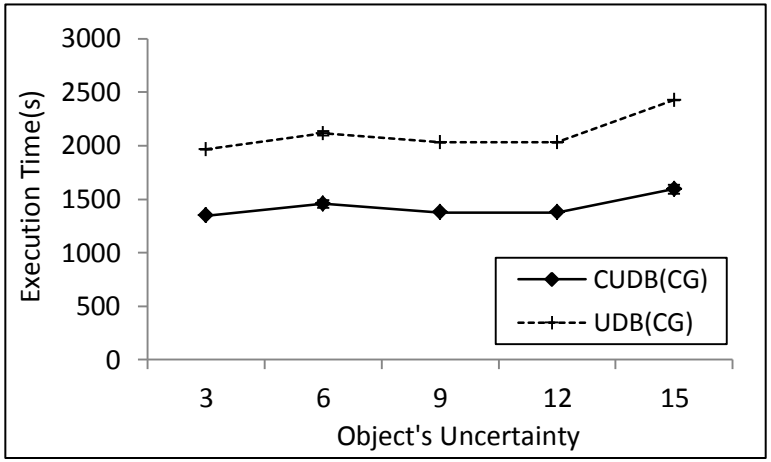
TG (synthetic dataset) $D = 100$, $\sigma = 5, t = 3, r = t\sigma$ and $p = 0.998$.

Efficiency Evaluation (2/2)

- Vary object's uncertainty (σ)
  - Increase in σ results in smaller probability $\Pr(o_i, o_j, D)$ values resulting in loose cell-bounds in CUDB(CG).



TG (synthetic dataset)



MOW (real dataset)

$D = 100,\ t = 3, r = t\sigma,\ \sigma_p = 4, \sigma_{SC} = 5$ and $p = 0.99$.

Outline

- Introduction and Background
- Problem Definition
- Outlier Detection on Uncertain Static Data (UDB)
- Continuous Outlier Detection on Uncertain Data Streams (CUDB)
- Experimental Evaluation
- Conclusion and Future Work

Conclusion and Future Work

Conclusion

- Proposed incremental outlier detection approach for uncertain data streams.
- Cell-based approach is utilized to prune cells containing only inliers or outliers.
- Experiments prove that the proposed approach is efficient and scalable.

Future Work

- Extend current work for higher dimensional data and other uncertainty models.

Thank you very much!

Salman Ahmed SHAIKH

Department of Computer Science
University of Tsukuba, Japan

A Framework of Faceted Search for XML data

Takahiro Komamizu

University of Tsukuba

02/19/2014

XML: Extensible Markup Language

- Semi-structured data, regarded as tree structured data.
- A de facto standard for exchanging data formats used in various fields.
  - Business data
    - ebXML[1], XBRL[2]
  - Bibliographic information
    - DBLP, SIGMOD records
  - Scientific data
    - Swiss-Prot, KEGG[3]
  - and so on.

[1] Electronic Business using XML
[2] Extensible Business Reporting Language
[3] Kyoto Encyclopedia of Genes and Genomes

Searching over XML data

- Searching methods:
  - Path-based search: XPath, XSLT, and XQuery.
  - Keyword-based search: LCA-based search.
- Users:

Experts of data.	Non-experts of data.
• deep knowledge about data (structures and contents)	• shallow knowledge about data
• precise queries	• vague queries

- Users may need systematic supports, esp. for non-experts, such as **exploratory search**, but exploratory search is not supported by conventional search methods.

Exploratory Search

- Trial-and-error manner
  - interleavingly repeat search by queries and judge results
  - search system gives hints to modify previous queries
- **Faceted search** is one of exploratory search methods
  - search by selecting/deselecting **facets** (or categories, attributes, dimensions) of data
  - system shows corresponding facets for every query results
  - real applications: Amazon, eBay, DBLP, IEEE Xplore, etc.
- Good chemistry with ad-hoc search

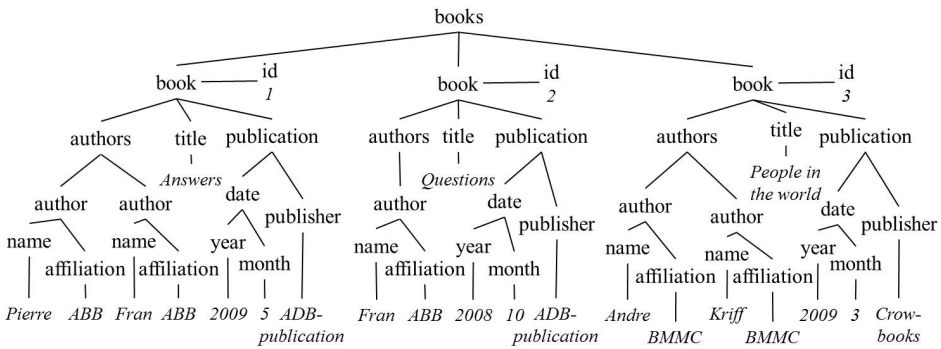
Research Direction

Apply faceted search for XML data

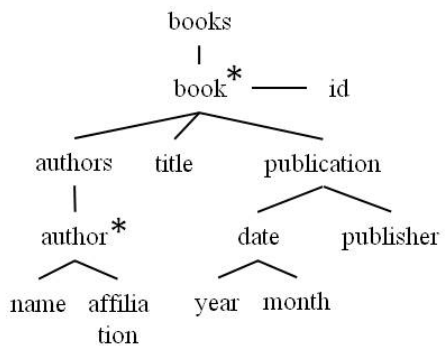
- Problems
  - objects are not defined in advance, because XML has complex structure comparing with records.
  - facets (or attributes of objects) are not defined, either.
- Proposal: a faceted search framework for XML data
  - extract objects and attributes using structural information.
  - propose three kinds of selection operations.
  - develop a framework which enables faceted search for given XML data.

Classes and Attributes

- From the structural information[4], frequently occurred XML elements below a parent element (*-ed in the figure) are regarded as **classes**.
- **Attributes** of an object are XML elements which are below the object element and directly contain text elements.



(a) XML data example.

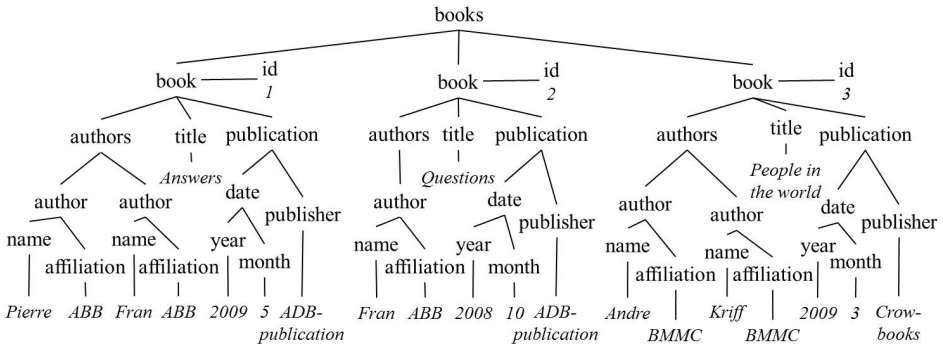


(b) Structural info.

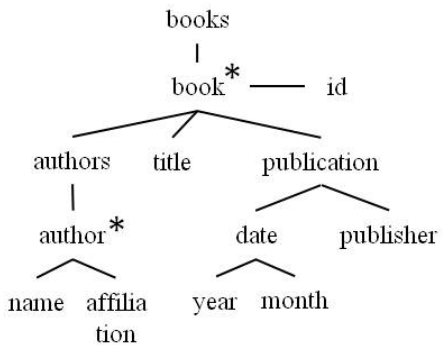
[4]If structural information (e.g., DTD and XML Schema) are not given, existing structural information extracting techniques (like DataGuide) can be

Objects and Facets

- An **object** is a tree rooted by an element regarded as a class.
- **Facets** are union set of all attributes belonging to objects.
- Values of facets are values of text elements.



(a) XML data example.

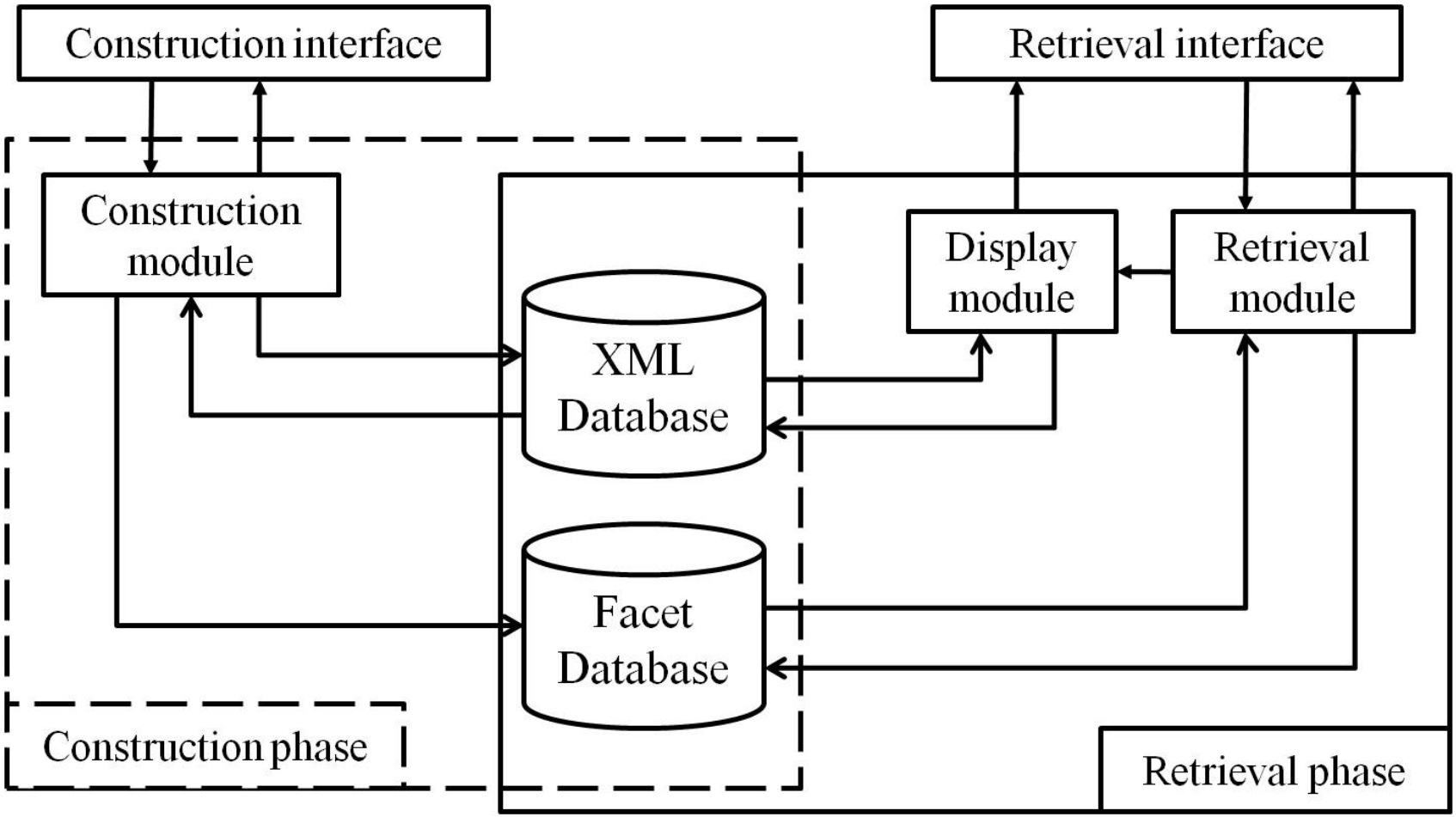


(b) Structural info.

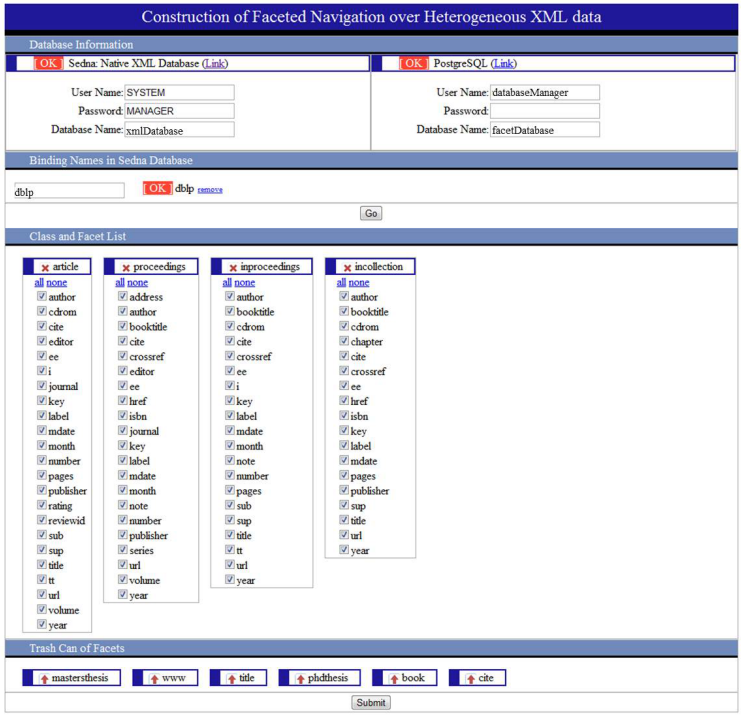
Operations

- Selection operation
  - find objects by selecting facet-value pairs
- Class-based selection operation
  - find objects by selecting classes
- Keyword-based selection operation
  - LCA-based keyword search s.t. results are objects which contain all input terms

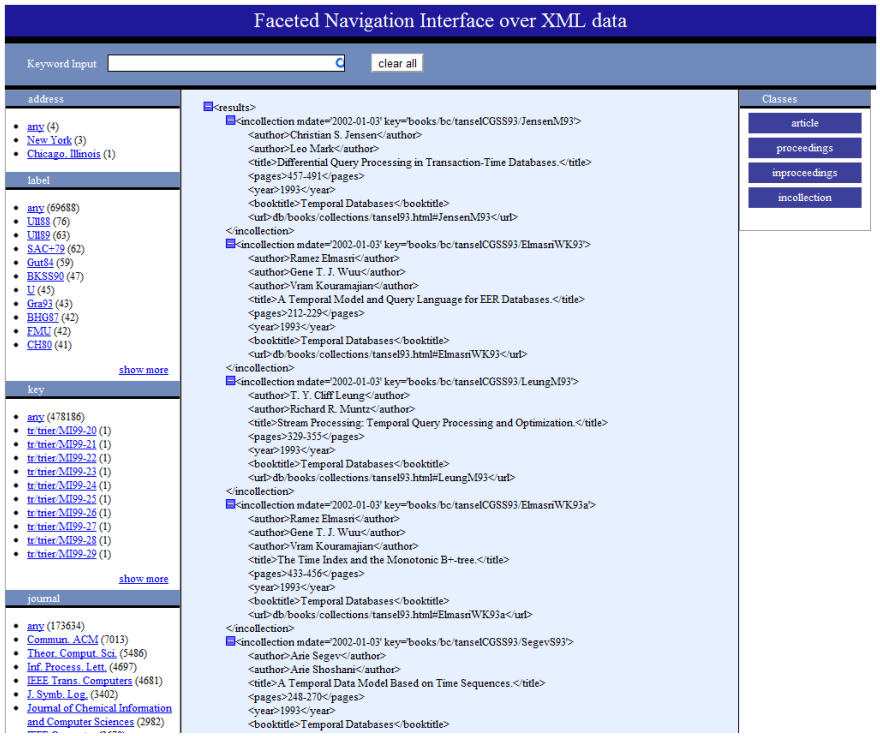
Framework Architecture



Construction Phase and Retrieval Phase



(a) Construction phase



(b) Retrieval phase

Overview of Experiments

- Purpose: check whether faceted search helps users to search over XML data
- Dataset: XML data of DBLP bibliography (200MB)
- Methodology: measuring average elapsed time for each task
- Examinees: 10 examinees
- Tasks: five tasks
  - Three exploratory task: having six characteristics (uncertainty, ambiguity, discovery, unfamiliar domain, low-level specificity, and imaginative situation)
  - Two ad-hoc query task: specific tasks

Task Examples

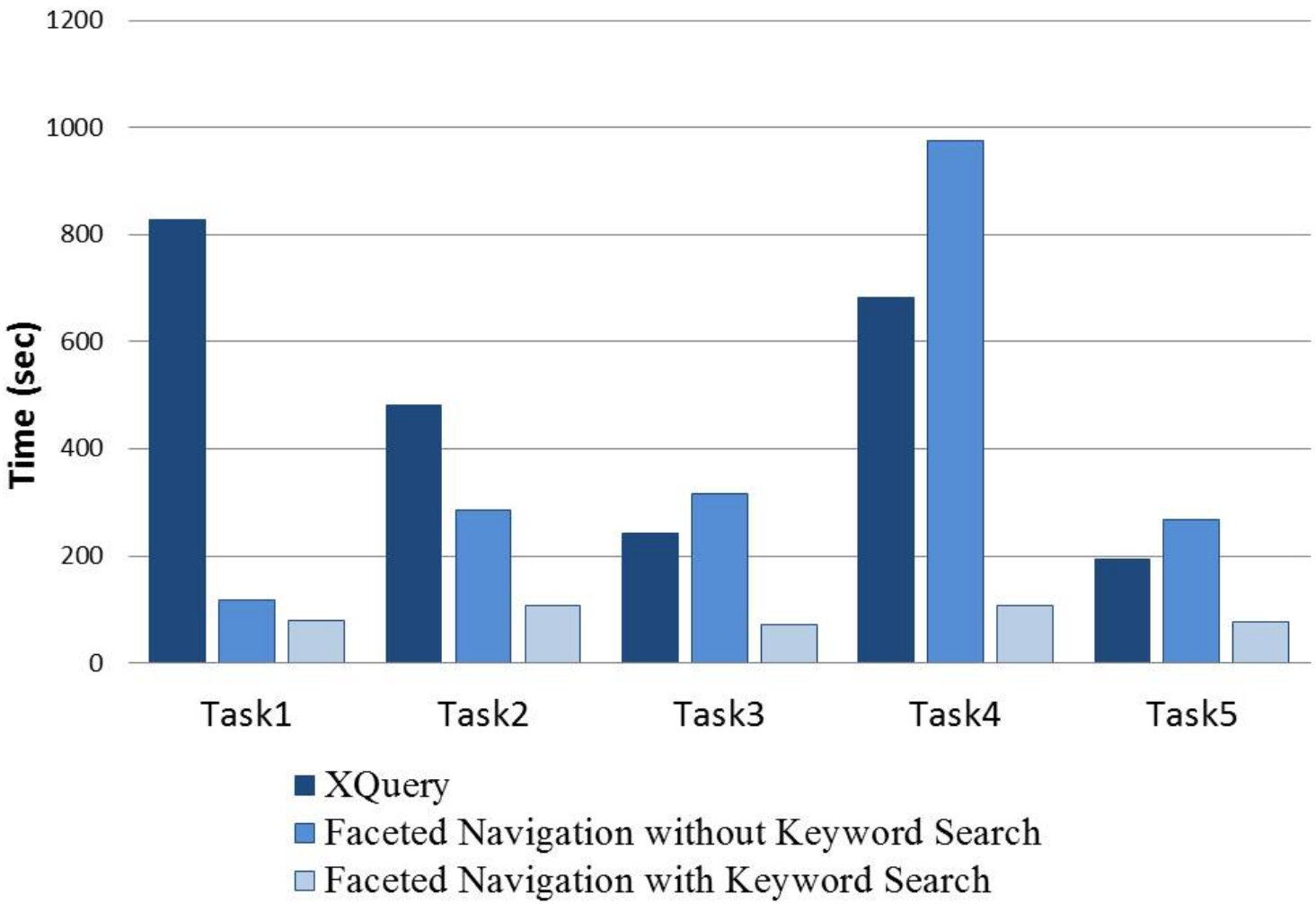
Exploratory task

Imagine a situation that one of your colleagues asks you about Michael J. Franklin. You just know his name, but do not any other information else, like details of his research topics and papers. You need to find three research topics of Michael J. Franklin.

Ad-hoc query task

Imagine that you are taking a course named "Systematic Languages" in that you learn several kinds of programming languages. From the next class, you will learn OCaml, and you are asked to read a paper entitled "Using, Understanding, and Unraveling the OCaml Language. From Practice to Theory and Vice Versa." Find this paper.

Experimental Results



Conclusion and Future works

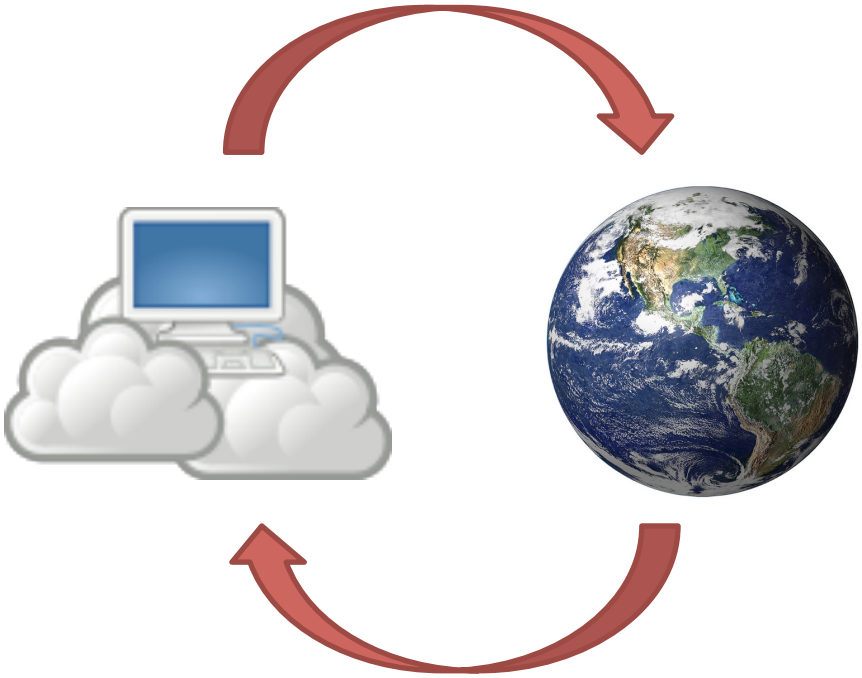
- Conclusion
  - Framework of faceted search over XML data
  - Experiments show applying faceted search for XML data helps users
- Future works
  - Extension for **graph data**
    - homogeneous graph like citation network
    - heterogeneous graph like paper-author network
  - Treatment for **longer texts**
    - if topics are given, topics as facets
    - otherwise, topics may come from topic models (e.g., LDA)
  - Ranking of results w.r.t. users' actions
    - order of selections of facets may be related to expectation for ranking

A Study on User Location Inference in Social Media

Yuto Yamaguchi
Ph.D student

Social Media Reflects the Real World

- ✓ Real-time
- ✓ Individuals
- ✓ Location information



Monitoring Real World

Event Detection [Sakaki+, 10] [Walther+, 13] …

- Earthquakes, typhoons, fires, ...

Analyzing Epidemics [Paul+, 11] [Aramaki+, 11] …

- Where and What

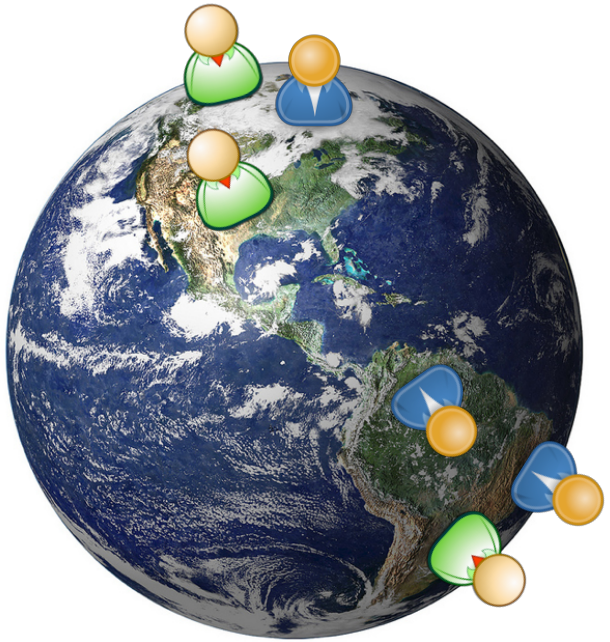
Disaster Analysis [Vieweq+, 10] [Mandel+, 12] …

- Situation Awareness

Importance of location information

Residential information is essential for monitoring the real world

- ✓ Where contents were posted
- ✓ Where events happened
- ✓ Where epidemic is



Our Problem

Location profiles are unavailable for …

- ✓ 74% of Twitter users [Cheng+, 10]
- ✓ 94% of Facebook users [Backstrom+, 10]

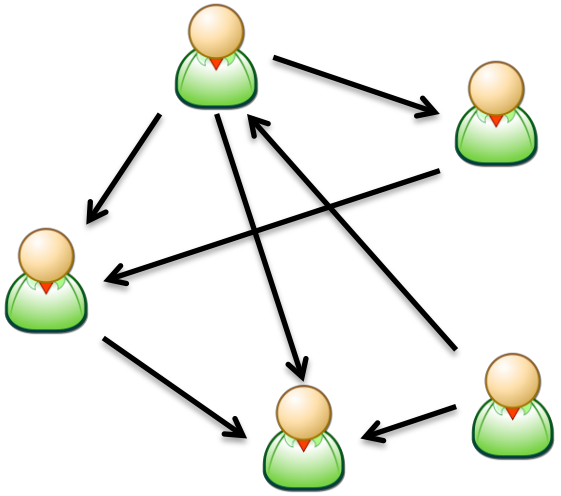
Why
- ✓ No merit
- ✓ Privacy

➔Worth inferring home locations

User Location Inference

Graph-based approaches

Analyzing social graphs
[Backstrom+, WWW'10]
[Rout+, HT'13]
[Jurgens+, ICWSM'13]



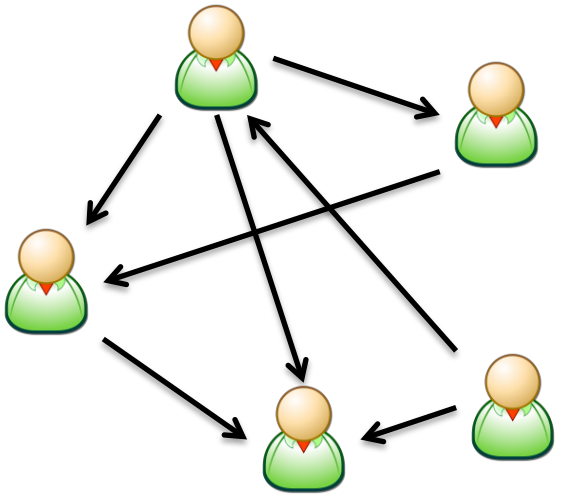
Content-based approaches

Analyzing user-generated contents
[Cheng+, CIKM'10]
[Hecht+, CHI'11]
[Kinsella+, SMUC'11]

User Location Inference

Graph-based approaches

Analyzing social graphs
[Backstrom+, WWW'10]
[Rout+, HT'13]
[Jurgens+, ICWSM'13]



This talk

Graph-based approaches (1/2)

Basic Idea

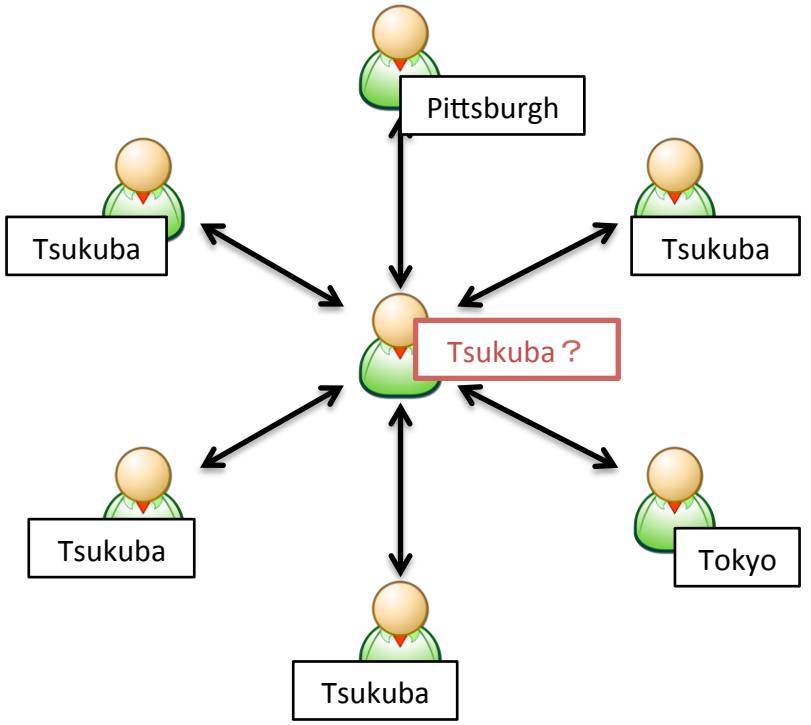
✓ Friends live close each other (***closeness assumption***)

Method

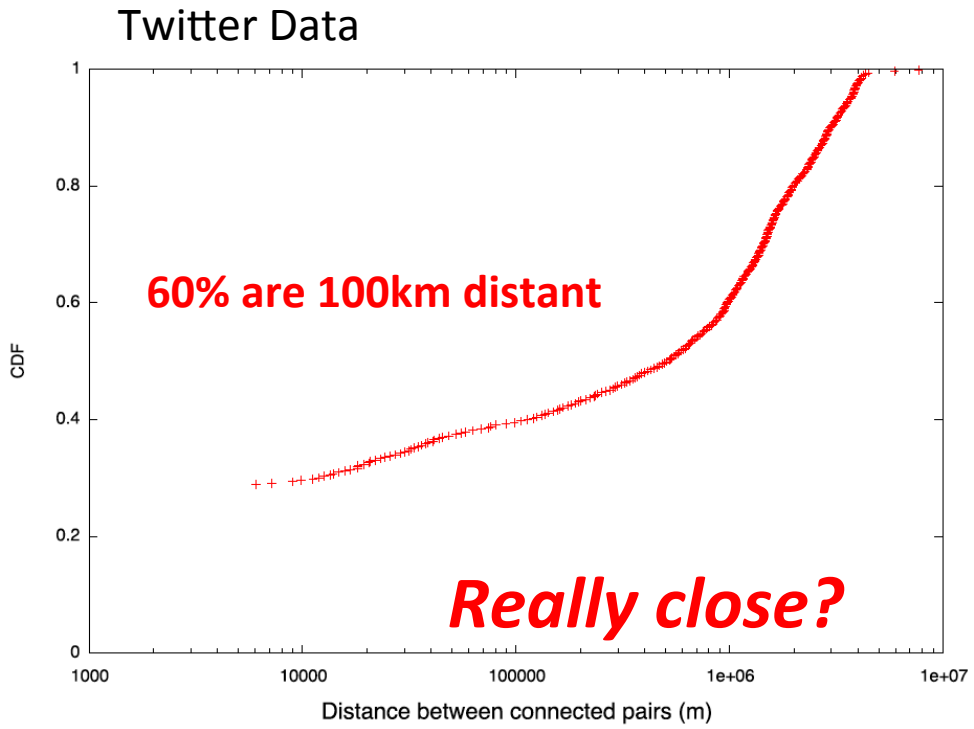
✓ Inferring the home location of the target user based on the home locations of friends

Roots

[Backstrom+, WWW'10]



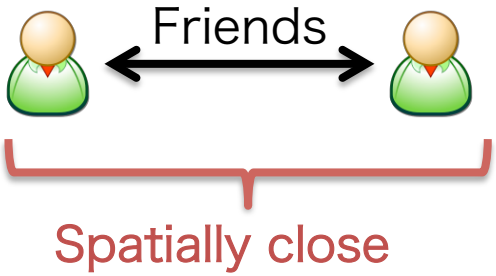
Graph-based approaches (2/2)



Concentration Assumption

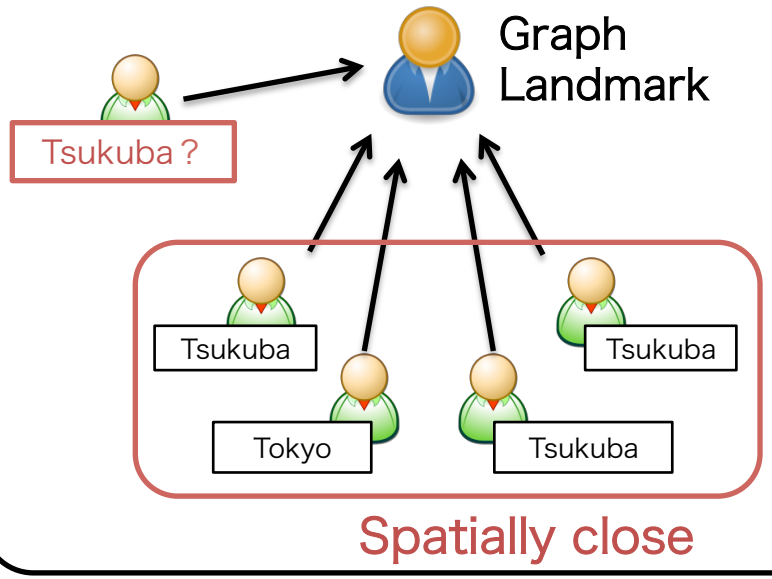
(Existing)
Closeness assumption

✓ IF friends
THEN close



(Proposed)
Concentration assumption

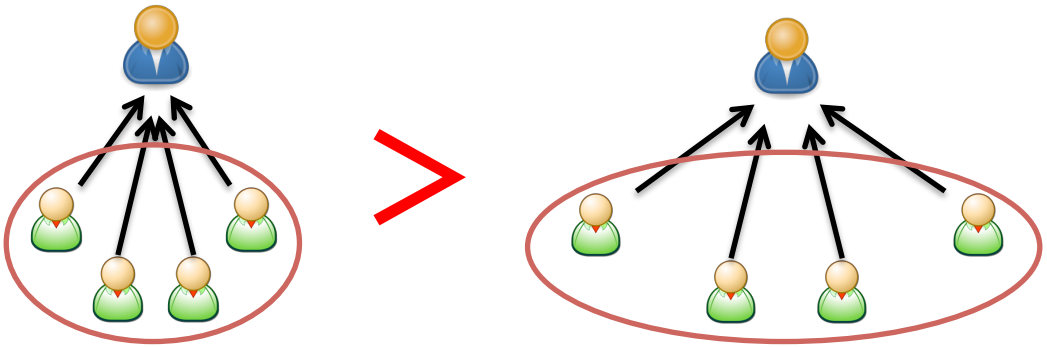
✓ IF follow ***graph landmarks***
THEN close



Required features of graph landmarks

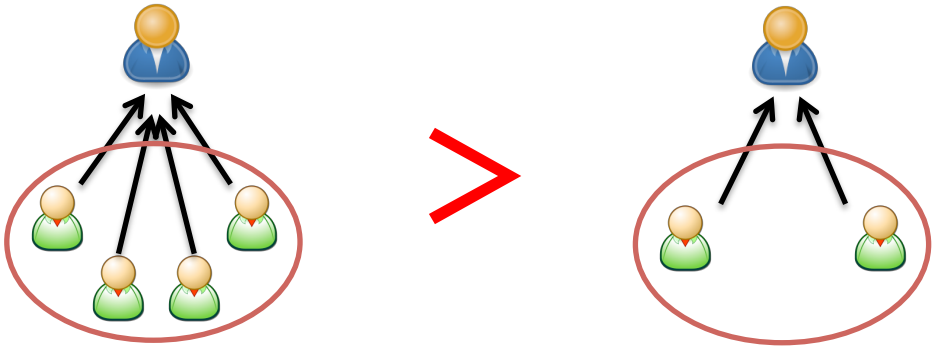
① **Small dispersion**

Followers are close to each other



② **Large degree**

A lot of followers



Examples in Twitter

Characteristics

- ✓ Tend to be non-humans
- ✓ Tend to post about its area
- ✓ Tend to be in metropolises

Overview

Probabilistic model

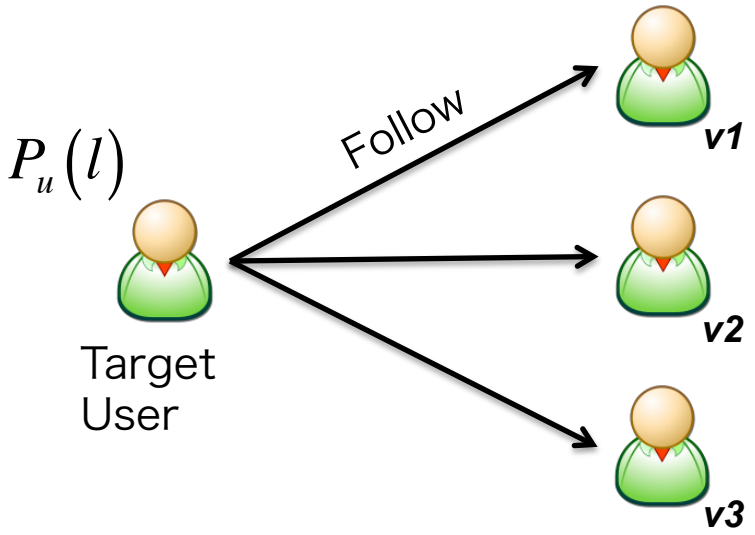
$$\hat{l}_u = \underset{l \in L}{\operatorname{argmax}} P_u(l)$$

$L \subset \mathbf{R}^2$: Geographical space

1. Model distribution P_u
2. Pick the mode point as the answer

➔ Landmark mixture model

Landmark Mixture Model (LMM)



	Dominance Distribution	Weights
v1	$D_{v1}(l)$	π_{v1}
v2	$D_{v2}(l)$	π_{v2}
v3	$D_{v3}(l)$	π_{v3}

$$P_u(l) = \pi_{v1} D_{v1}(l) + \pi_{v2} D_{v2}(l) + \pi_{v3} D_{v3}(l)$$

Mixture of distributions of v1, v2, and v3

Dominance Distributions

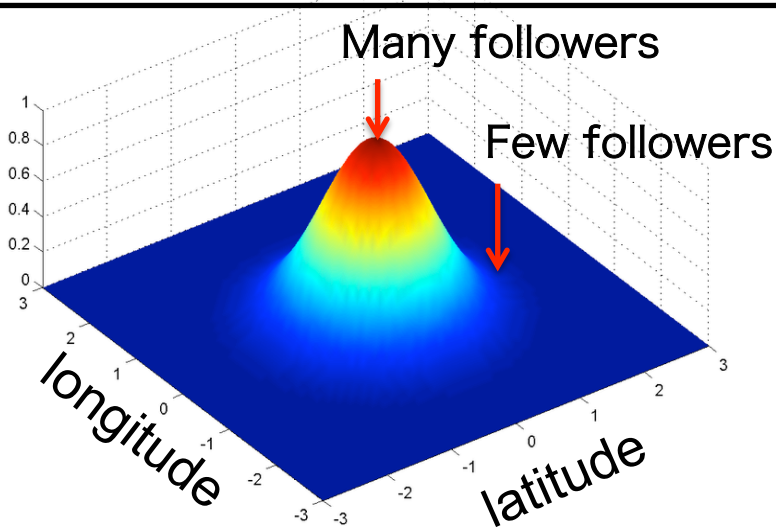
Spatial distributions of followers' home locations

➔ Gaussian

$$D_u(l) = N\left(l \middle| \mu_u, \Sigma_u\right)$$

Graph landmarks have small covariances

- Sharp
- Strong clues



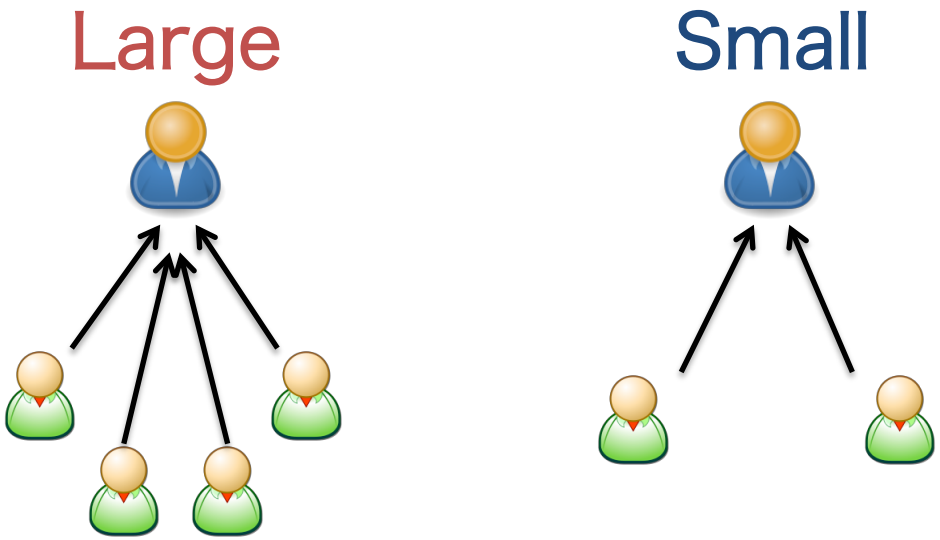
"Followers of this user are likely to live around here"

➔ Take into account **"Small dispersion"**

Mixture Weights

Proportional to the logarithm of in-degrees

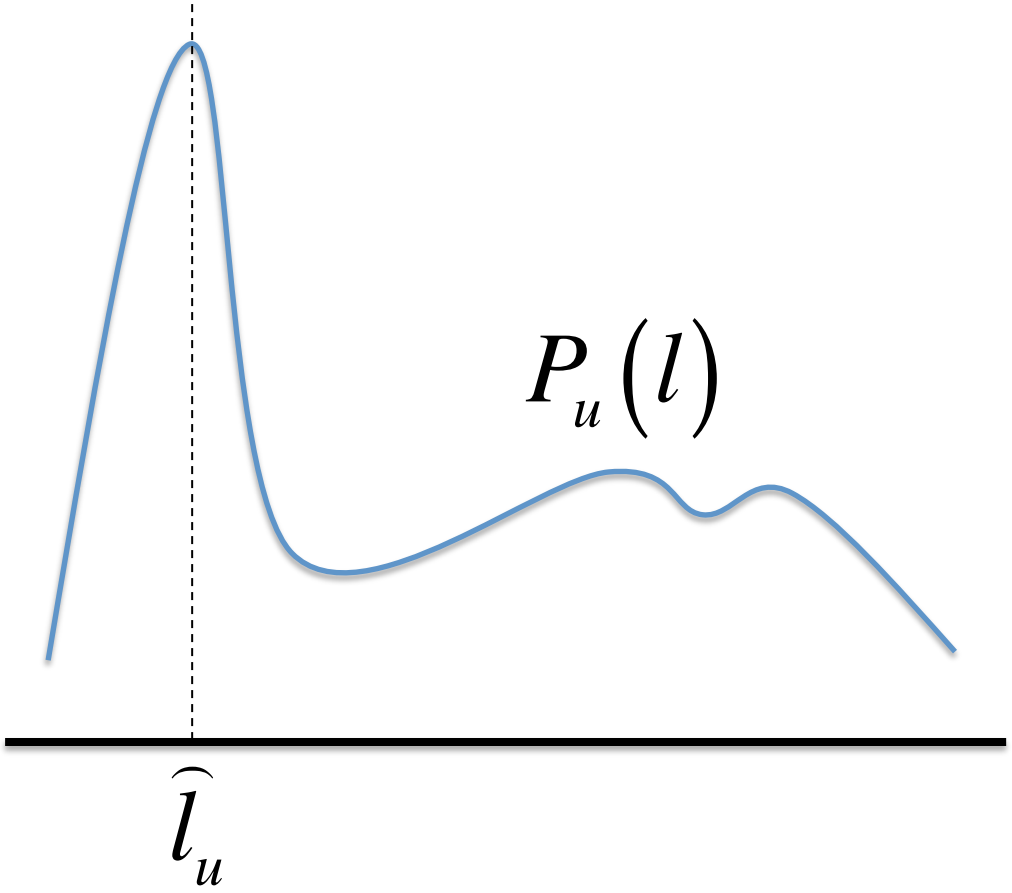
$$\pi_v \propto \log c_v$$



➔ Take into account “Large degree”

Mode point

Pick the mode point as the answer



Dataset

Twitter dataset [Li+, KDD'12]

- ✓ 3M users in the U.S.
  - ✓ 465K labeled users（15% of all users）
  - ✓ 46K users for test set (10% of labeled users)
- ✓ 285M follow edges

Evaluation

1. Infer home locations of test users by compared methods
2. Calculate the error distance between inferred locations and true locations

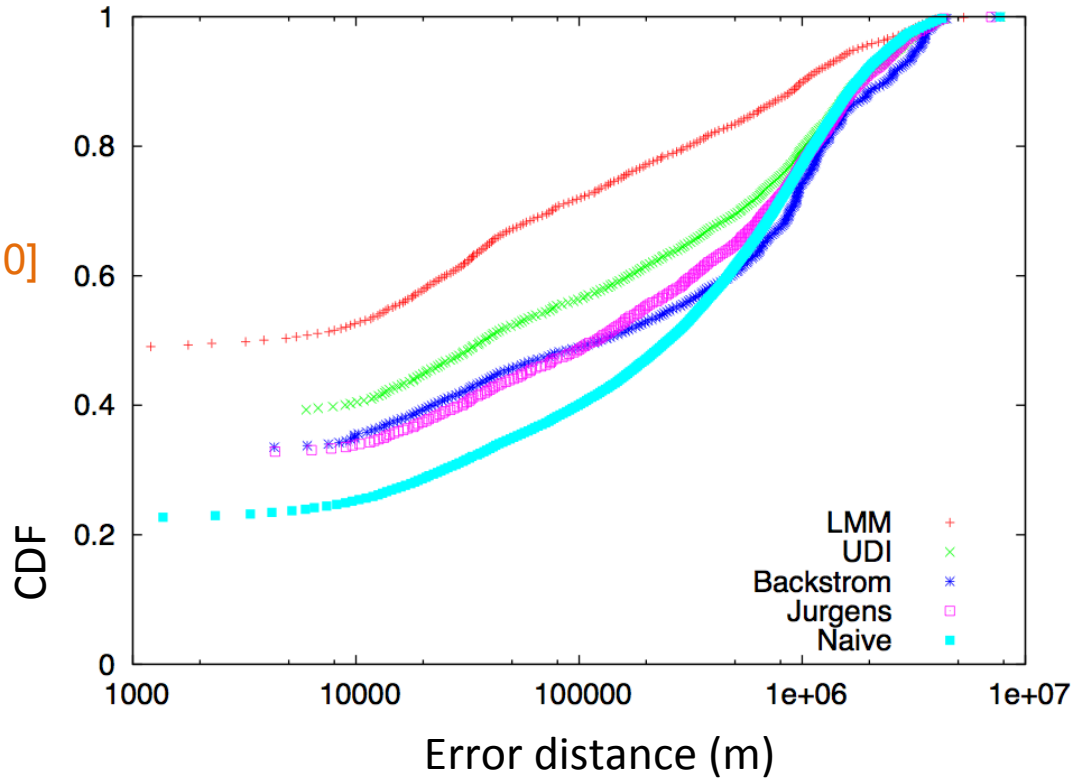
Performance comparison

Compared methods

- ✓ LMM: Our method
- ✓ UDI: [Li+, KDD'12]
- ✓ Backstrom: [Backstrom+, WWW'10]
- ✓ Jurgens: [Jurgens, ICWSM'13]
- ✓ Naïve: Medoids

Results

- ✓ Best by LMM



Contributions

① Introduced **concentration assumption**

Exploiting **graph landmarks**

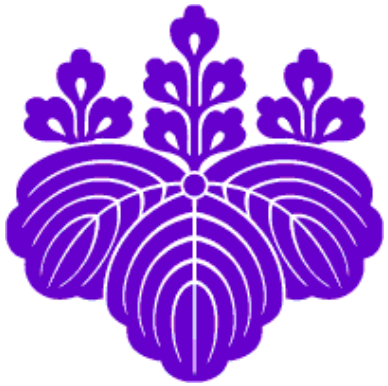
② Proposed **Landmark mixture model**

More accurate than existing methods

◆ **Publication**
Landmark-Based User Location Inference in Social Media
Yuto Yamaguchi, Toshiyuki Amagasa, and Hiroyuki Kitagawa
The 1st ACM Conference on Online Social Networks (COSN 2013), pp.223-234, Boston, USA, October 7-8, 2013.

An Efficient Sorting Algorithm on GPUs

Yusuke Kozawa
Toshiyuki Amagasa
Hiroyuki Kitagawa



筑波大学
University of Tsukuba

Background

- GPU (Graphics Processing Unit)
  - Have multiple cores
    - Each core has tens to hundreds of scalar processors
  - High performance and high memory bandwidth
  - ***GPU computing***

- Sorting
  - Fundamental operation
  - Its GPU acceleration has been widely studied

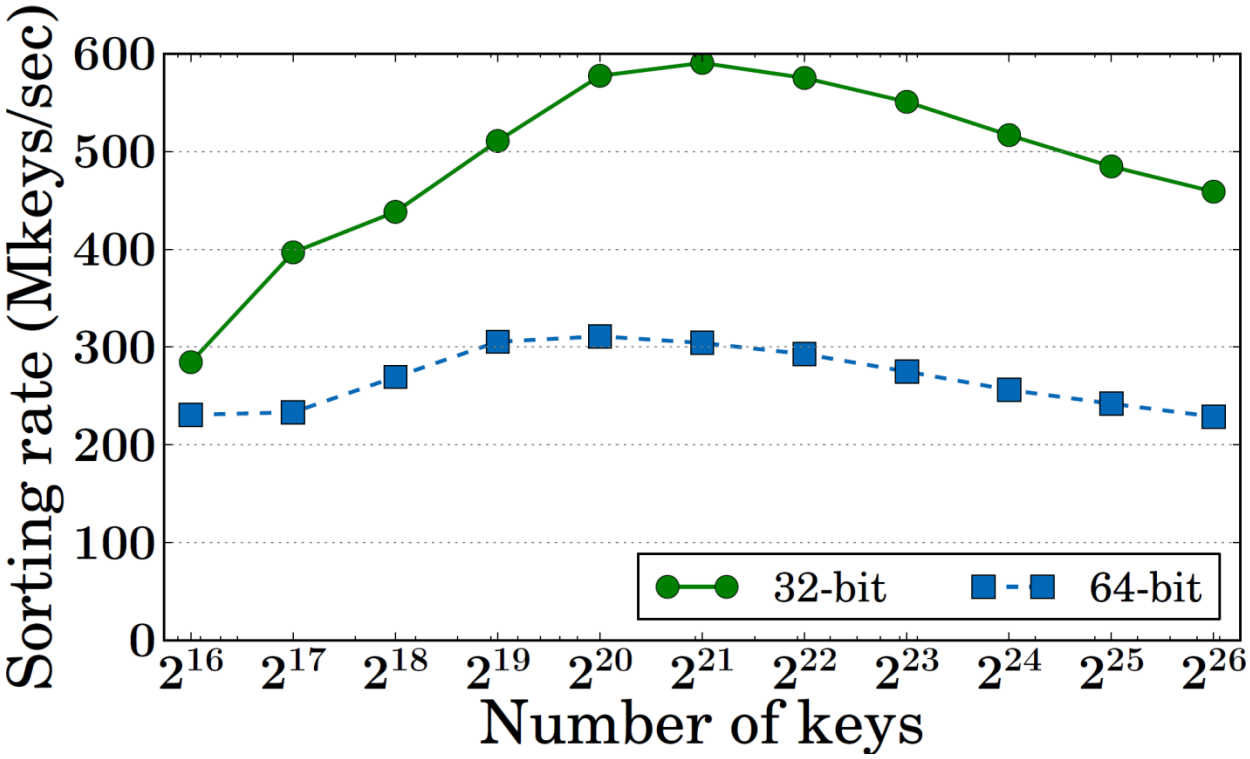
Sorting on GPUs

- Radix sort [Merrill+, PPL '11]
  - Non-comparison sort
  - Pros: Low time complexity $O(kn)$
  - Cons: Complexity depends on key length k

- Merge sort [Baxter, '14]
  - Representative of comparison-sort algorithms
  - Pros: Complexity is independent on key length
  - Cons: high time complexity $O(n\log n)$

The Performance of Merge Sort

- Performance degrades when data is large
  - Bounded by memory bandwidth



Our Contributions

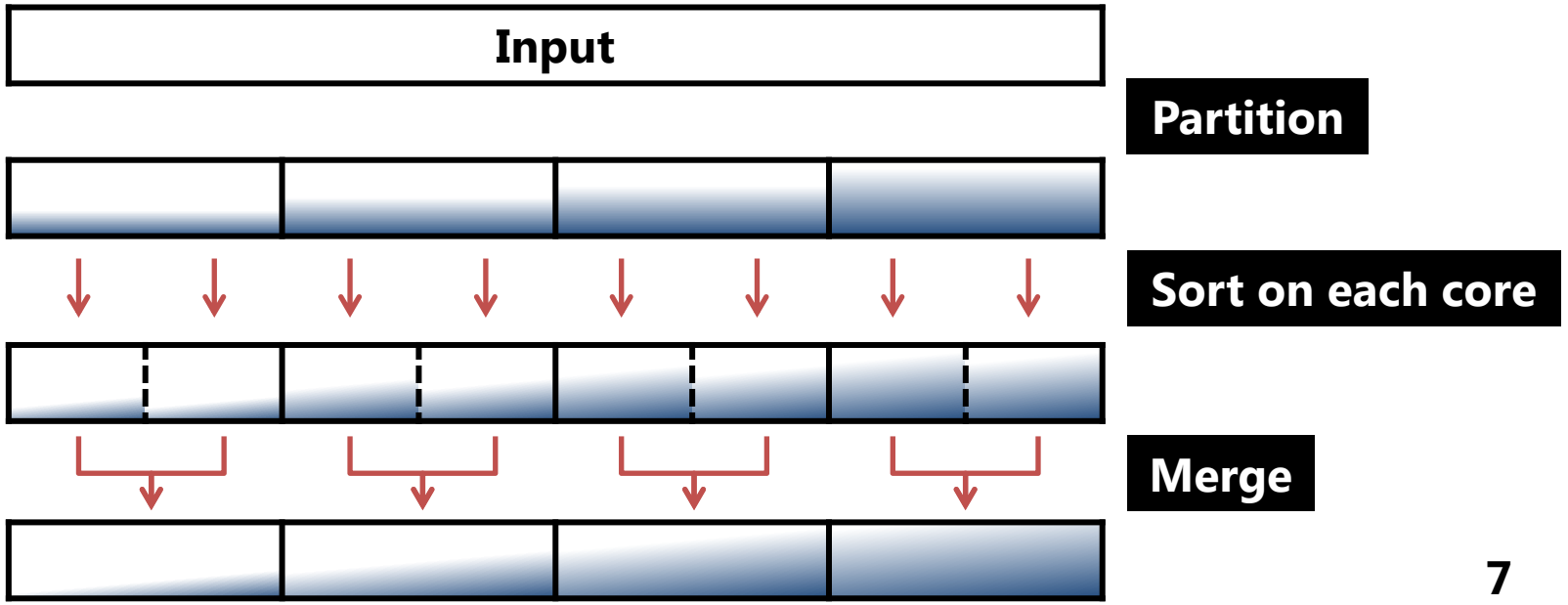
- Propose a sorting algorithm on GPUs
  - Reduce memory accesses by partitioning data based on samplesort [Leischner+, IPDPS '10]
  - Sort the partitioned data by using the merge algorithm of Baxter's merge sort
- Evaluate the algorithm by experiments
  - Up to 1.39 times faster than Baxter's merge sort
  - Similar performance to radix sort for 32-bit keys
    - Up to 2.3 times faster than radix sort for 64-bit keys

Outline

- ✓ Background and Contributions
2. Proposed Algorithm
3. Experiments
4. Conclusion and Future Work

An Overview of Our Algorithm

- Basic idea
  - Lower the number of merges by partitioning the data into ***k buckets***
  - Cooperatively merge the data within buckets with ***a single kernel***



Data Partitioning

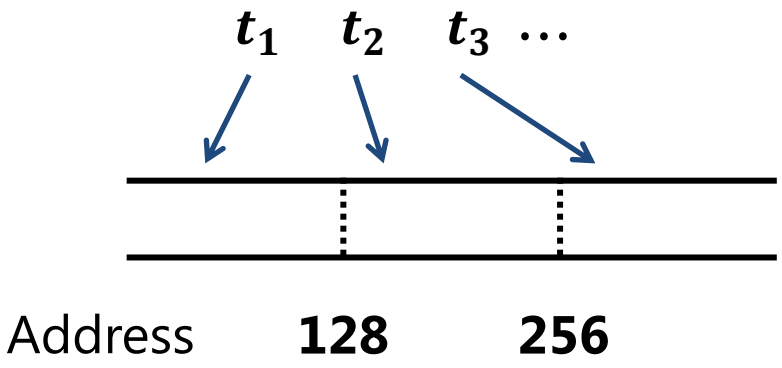
- Algorithm [Leischner+, IPDPS '10]
  1. Obtain $k-1$ splitters from the input
  2. Compute the scatter positions of the input data
     a. Count the sizes of buckets
     b. Calculate the offsets of buckets
  3. Data relocation

- Problem
  – When accessing buckets, memory access patterns do not meet the condition of ***coalesced accesses***

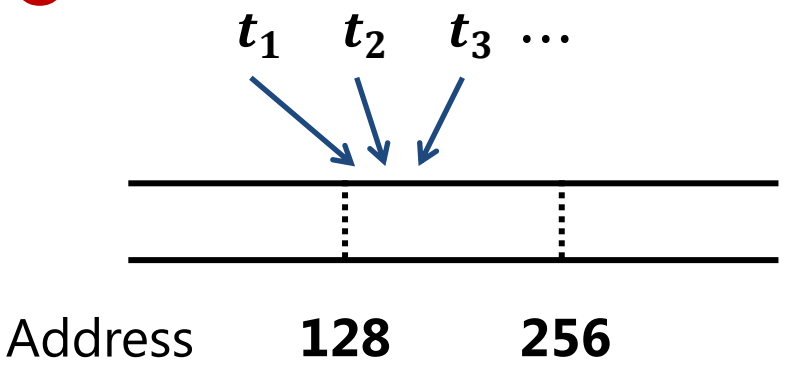
Coalesced Accesses

- Memory accesses can be coalesced into one
  - If a group of 32 threads accesses an aligned 128-byte region

X



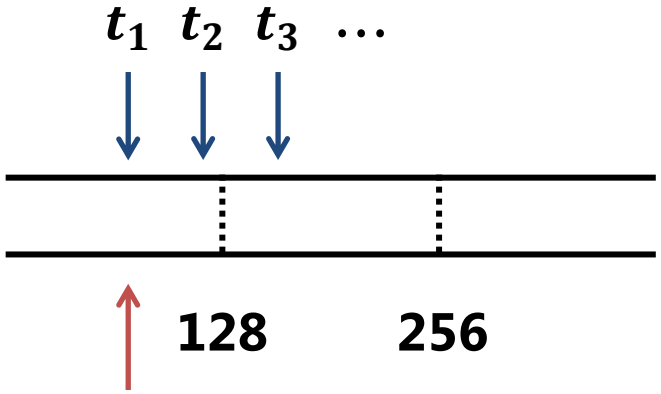
O



Optimize Memory Accesses

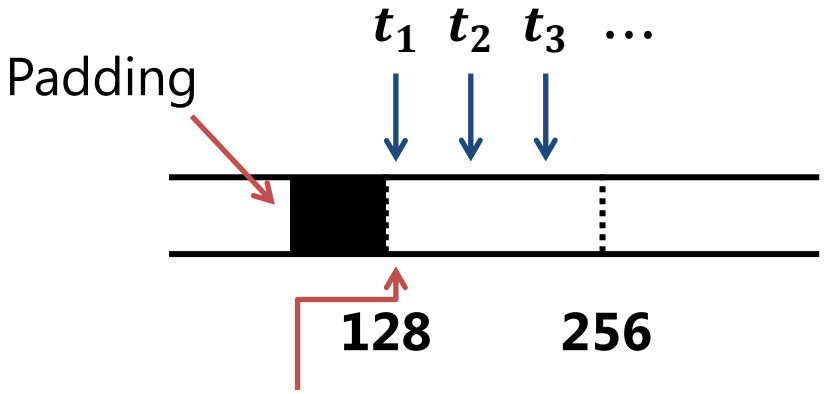
- ***Bucket alignment***
  - Align the starting position of each bucket
  - This makes the memory accesses to buckets meet the condition of coalesced accesses

Not aligned



Starting position of a bucket

Aligned



Starting position of a bucket

Sort the Buckets

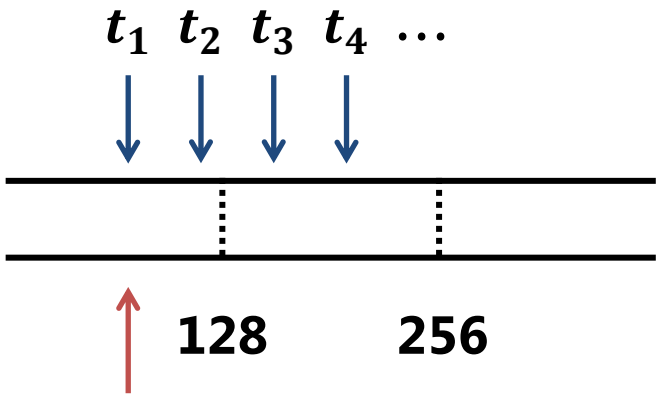
- Existing schemes for sorting buckets
  - Sort a bucket on a core [Leischner+, IPDPS '10]
  - Sort a bucket with a kernel [Dehne+, PPL '12]
  - Problems: Load imbalance, not enough parallelism

- ***Cooperative merge***
  - Merge the data within ***all buckets*** with ***a kernel***
  - Continue the merge until all the buckets are sorted

Restore the Positions of Buckets

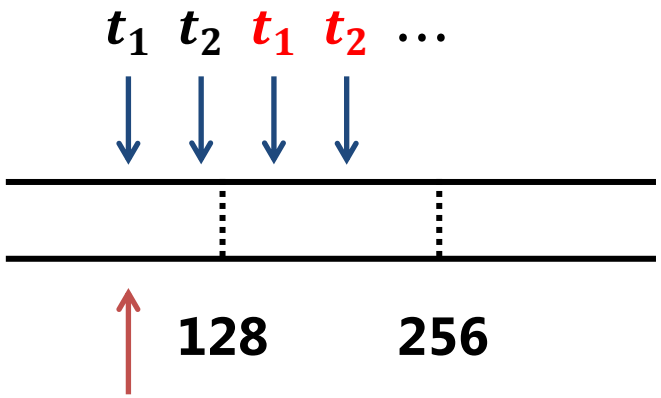
- In the last merge
  - Need to restore the positions of buckets
  - To coalesced memory accesses, separately handle the left and right of a 128-byte boundary

Not coalesced



Starting position of a bucket

Coalesced



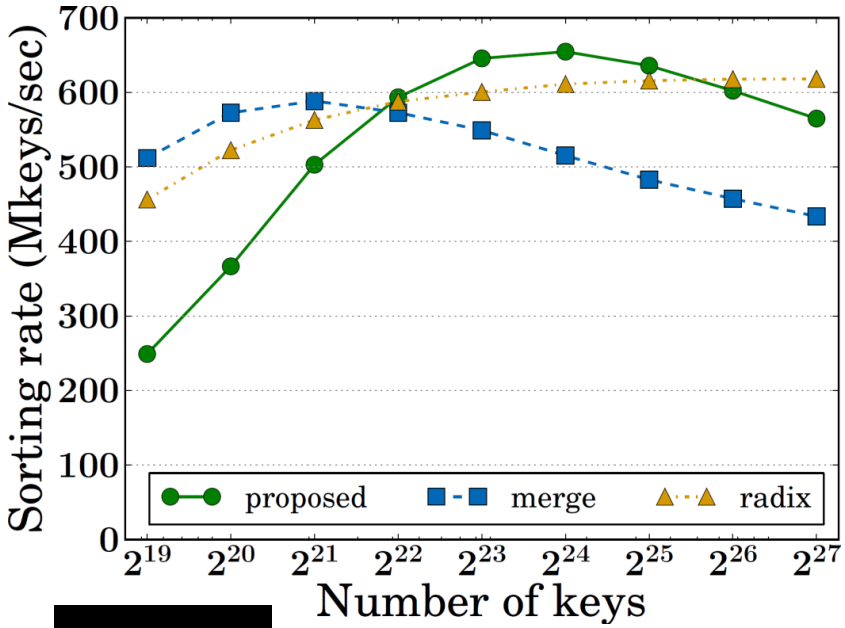
Starting position of a bucket

Experiments

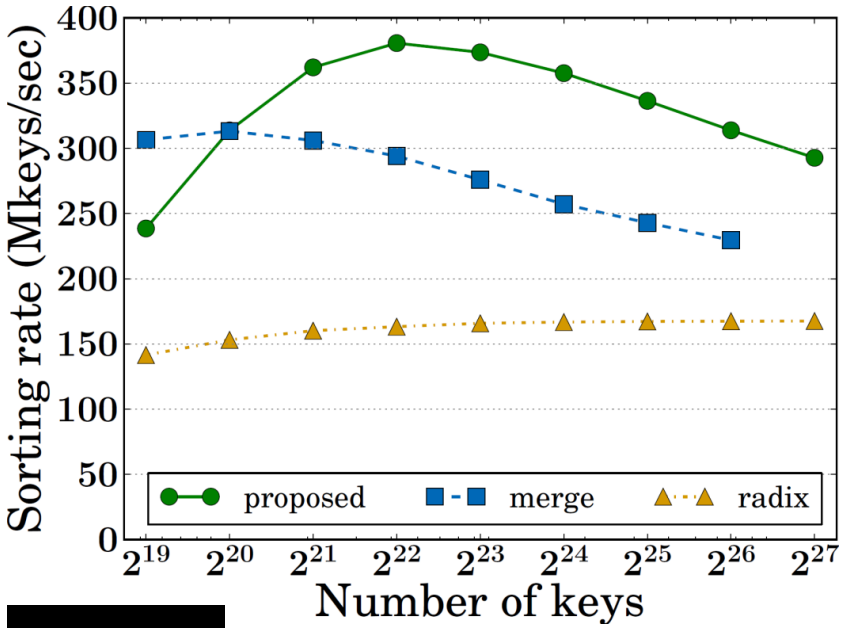
- Compared implementations
  - Merge sort [Baxter, '14]
  - Radix sort [Merrill, '13]
- Data
  - Arrays of 32-bit keys and 64-bit keys, respectively
  - Generated by a uniform distribution
- Machine configuration
  - OS: Ubuntu 12.04.3 LTS 64 bit
  - CUDA 5.5
  - GPU: Tesla C2050

Results: Comparisons

- Ours is up to 1.39 times faster than merge sort
- Compared with radix sort
  - Depends on the data size for 32-bit keys
  - Generally faster when 64-bit keys are used



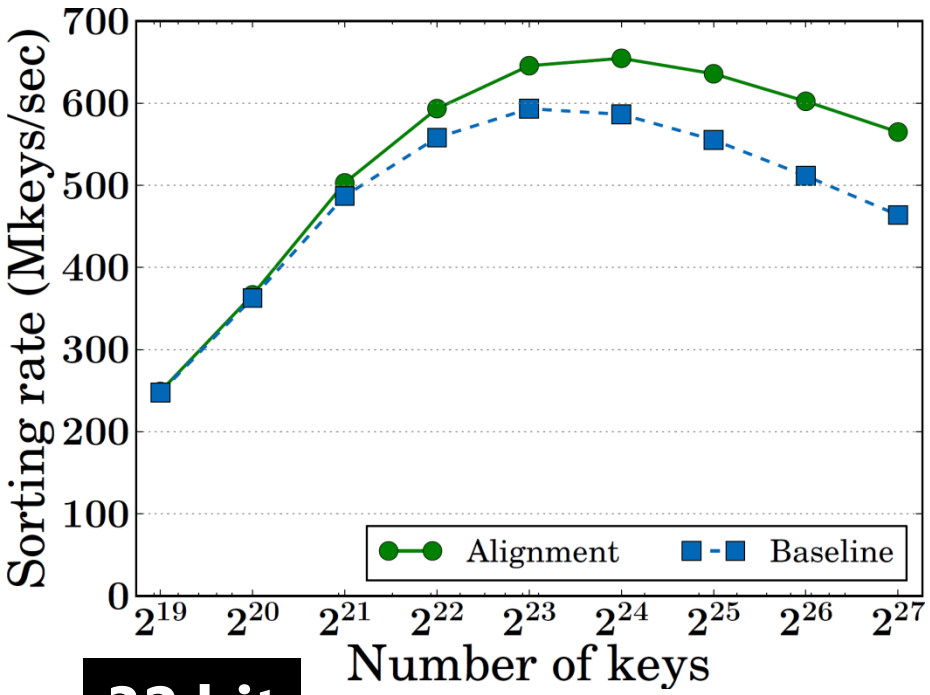
32 bit



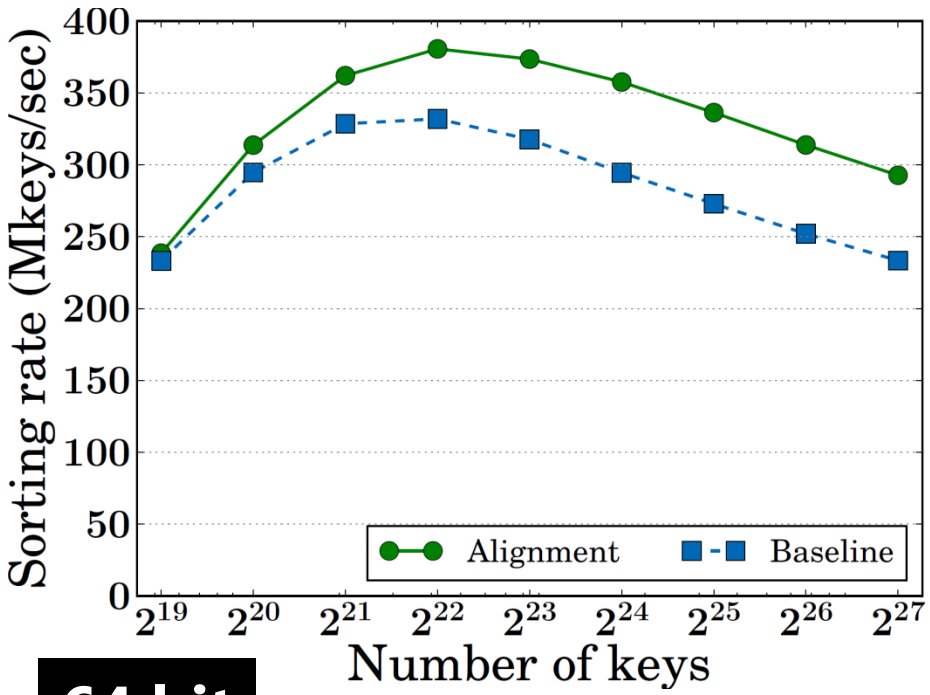
64 bit

Results: Effects of Optimization

- Speedup of up to 1.25
  - Effective especially for large data sizes



32 bit



64 bit

Conclusion and Future Work

- Conclusion
  - Propose a sorting algorithm on GPUs
    - Reduce memory accesses by data partitioning
    - Cooperatively merge buckets with load balance
  - Evaluate the algorithm by experiments
    - Faster than merge sort when the data is large enough
    - Similar or higher performance than radix sort
- Future work
  - Evaluation by using Kepler GPUs
  - Partition data more than once

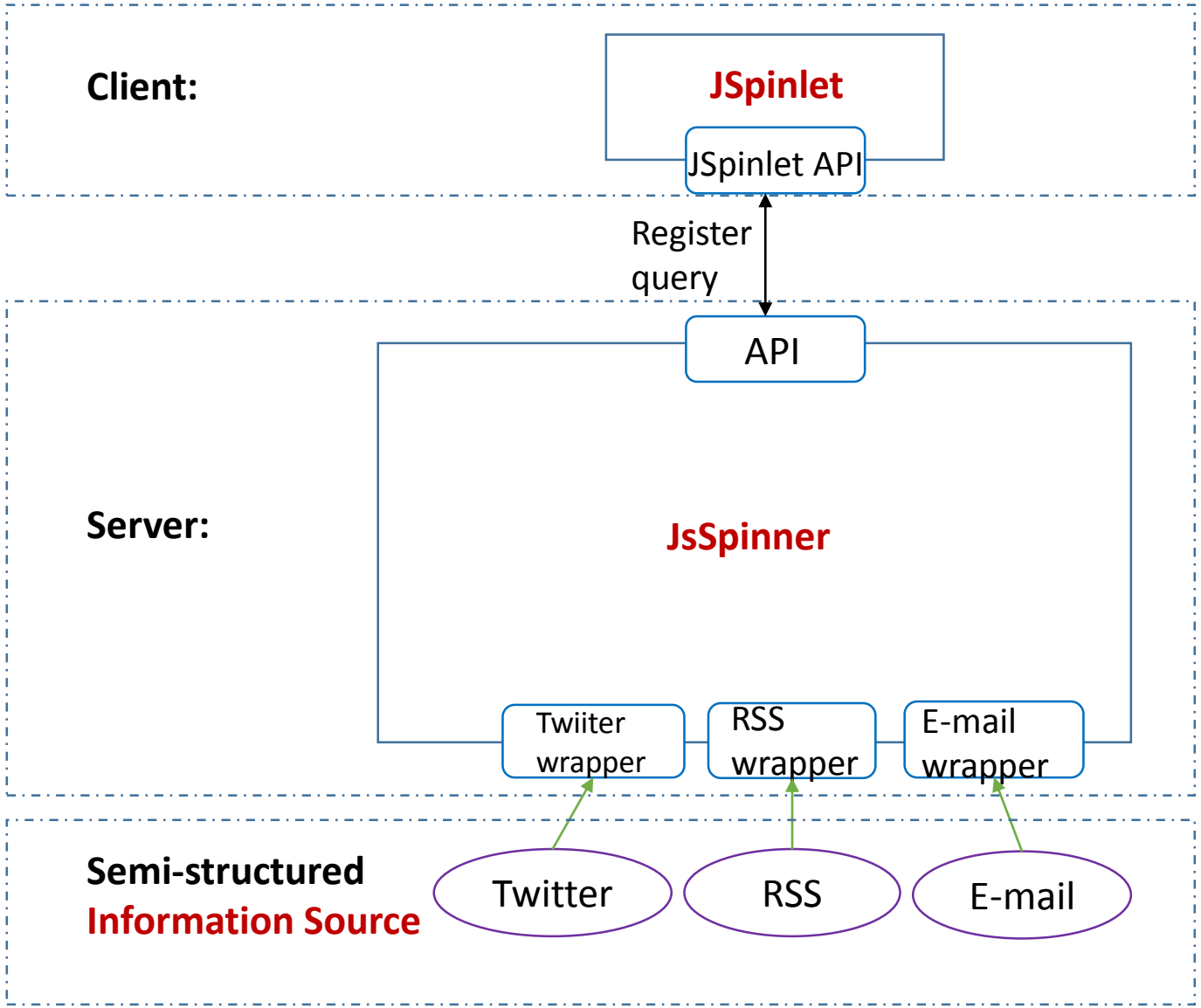
Introduction of JsSpinner

Wang Yan
2014-02-19

Stream processing engine

- Stream
  - Continuously generated data
  - Example: Tweets, stock trades, network packets
- Stream processing engine
  - Query stream data in real time
  - Continuous query can be registered
  - Whenever data comes, the query is processed.

System architecture

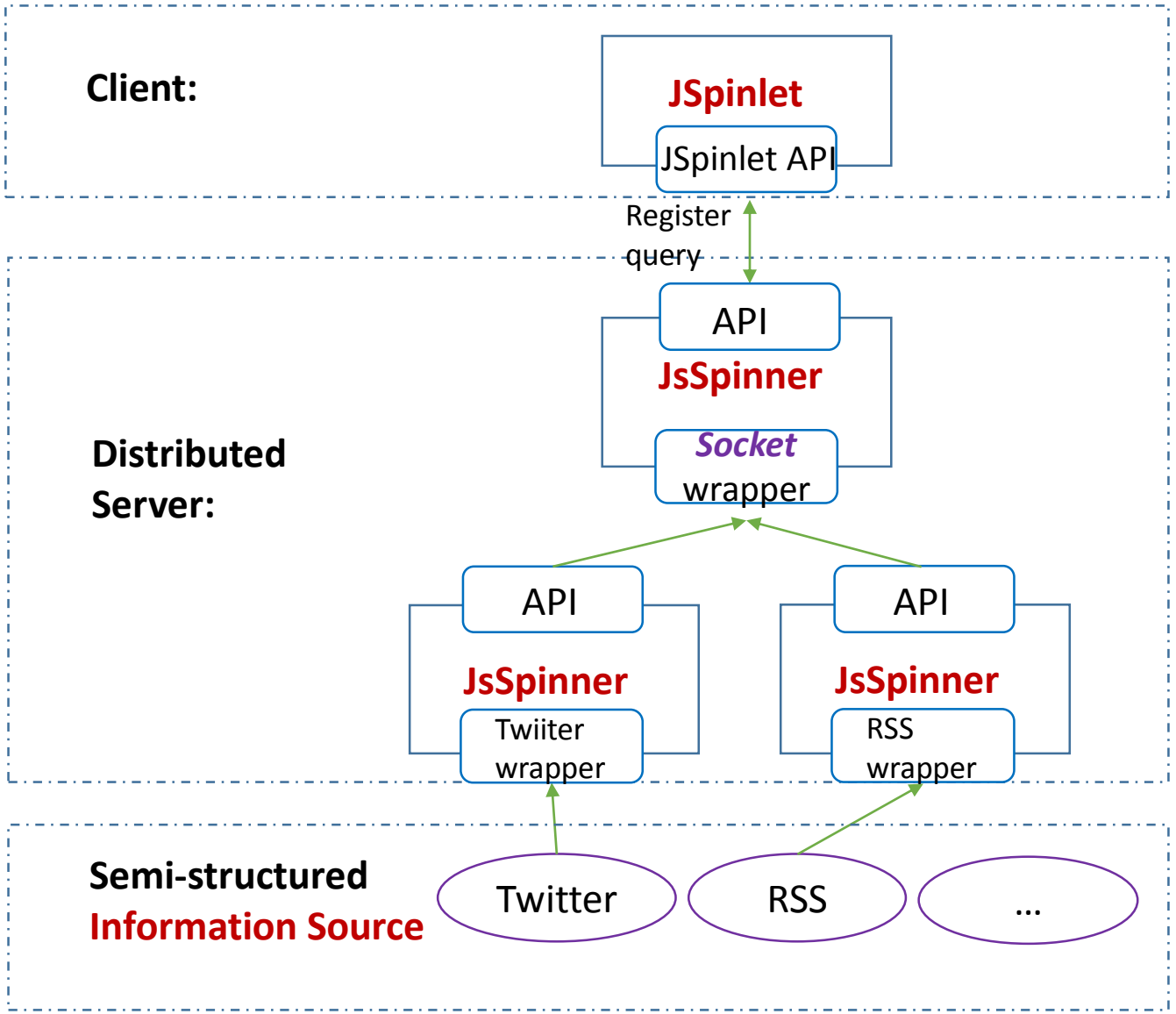


JSpinlet is client library

JsSpinner processes the query.

Wrapper is responsible for getting input data

Distributed stream processing



Query processing scheme

- Designated event-based processing
  - Users can specify some input streams as **master streams.**
  - The query is supposed to be processed and generates query results only when data comes from master streams.

Data model and operator

- JSON
  - semi-structured
  - lightweight
  - flexible

- Operator
- Selection
- Projection
- Join
- Groupby_aggregation
- Istream
- Dstream
- Rstream
- Expand

Demo---schema

```
{"id": "stream1",
       "type":"object",
       "properties":
           {
           "id":{
               "type": "string"
                },
           "student":
                {
               "type":"object",
               "properties":{
                   "age":{
                         "type":"number"
                         }
                             }
                }
           }
}
```

```
{"id": "stream2",
 "type":"object",
 "properties": {
                "id":{
                     "type": "string"
                     },
                "score":{
                     "type":"integer"
                        }
                }
 }
```

Demo--Query

Extension of Jaql

```
stream1 = readFromWrapper ("stream1", false) ;

stream2 = readFromWrapper ("stream2", true) ;

tmp1 = stream1 -> window[rows 2] ->  transform {$.id,$.student.age};

tmp2 = stream2 -> window[rows 3] ->  filter $.score >= 22;

j = join s in tmp1,
         d in tmp2
    where s.id == d.id
    into {s.id, d.score};

j -> istream;
```

Thank you very much

2014.02.19