

Agenda

1. **SIGMOD 2010 Paper**: Optimizing Content Freshness of Relations Extracted From the Web Using Keyword Search
2. **Cloud-based Parallel DBMS**
3. **Mining Workflows for Data Integration Patterns**
4. **Energy Efficient Complex Event Processing**



Optimizing Content Freshness of Relations Extracted From the Web Using Keyword Search

Mohan Yang (Shanghai Jiao Tong University),

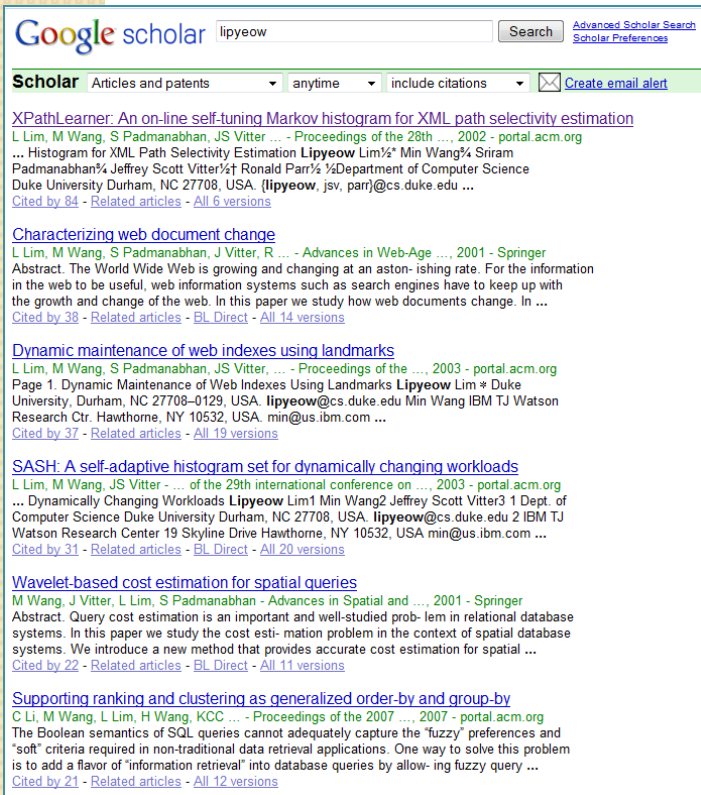
Haixun Wang (Microsoft Research Asia),

Lipyeow Lim (UHM)

Min Wang (HP Labs China)

Motivating Application

- Management at a prominent research institute wanted to analyze the impact of the publications of its researchers ...



Google scholar lipyeow Search Advanced Scholar Search Scholar Preferences

Scholar Articles and patents anytime include citations Create email alert

[XPathLearner: An on-line self-tuning Markov histogram for XML path selectivity estimation](#)
 L Lim, M Wang, S Padmanabhan, JS Vitter ... - Proceedings of the 28th ..., 2002 - portal.acm.org
 ... Histogram for XML Path Selectivity Estimation **Lipyeow Lim** Min Wang Sriram Padmanabhan Jeffrey Scott Vitter Ronald Parr Department of Computer Science Duke University Durham, NC 27708, USA. (lipyeow, jsv, par)@cs.duke.edu ...
 Cited by 84 - Related articles - All 6 versions

[Characterizing web document change](#)
 L Lim, M Wang, S Padmanabhan, J Vitter, R ... - Advances in Web-Age ..., 2001 - Springer
 Abstract. The World Wide Web is growing and changing at an astonishing rate. For the information in the web to be useful, web information systems such as search engines have to keep up with the growth and change of the web. In this paper we study how web documents change. In ...
 Cited by 38 - Related articles - BL Direct - All 14 versions

[Dynamic maintenance of web indexes using landmarks](#)
 L Lim, M Wang, S Padmanabhan, JS Vitter, ... - Proceedings of the ..., 2003 - portal.acm.org
 Page 1. Dynamic Maintenance of Web Indexes Using Landmarks **Lipyeow Lim** * Duke University, Durham, NC 27708-0129, USA. lipyeow@cs.duke.edu Min Wang IBM TJ Watson Research Ctr. Hawthorne, NY 10532, USA. min@us.ibm.com ...
 Cited by 37 - Related articles - All 19 versions

[SASH: A self-adaptive histogram set for dynamically changing workloads](#)
 L Lim, M Wang, JS Vitter ... of the 29th international conference on ..., 2003 - portal.acm.org
 ... Dynamically Changing Workloads **Lipyeow Lim** Min Wang Jeffrey Scott Vitter 1 Dept. of Computer Science Duke University Durham, NC 27708, USA. lipyeow@cs.duke.edu 2 IBM TJ Watson Research Center 19 Skyline Drive Hawthorne, NY 10532, USA min@us.ibm.com ...
 Cited by 31 - Related articles - BL Direct - All 20 versions

[Wavelet-based cost estimation for spatial queries](#)
 M Wang, J Vitter, L Lim, S Padmanabhan - Advances in Spatial and ..., 2001 - Springer
 Abstract. Query cost estimation is an important and well-studied problem in relational database systems. In this paper we study the cost estimation problem in the context of spatial database systems. We introduce a new method that provides accurate cost estimation for spatial ...
 Cited by 22 - Related articles - BL Direct - All 11 versions

[Supporting ranking and clustering as generalized order-by and group-by](#)
 C Li, M Wang, L Lim, H Wang, KCC ... - Proceedings of the 2007 ..., 2007 - portal.acm.org
 The Boolean semantics of SQL queries cannot adequately capture the "fuzzy" preferences and "soft" criteria required in non-traditional data retrieval applications. One way to solve this problem is to add a flavor of "information retrieval" into database queries by allowing fuzzy query ...
 Cited by 21 - Related articles - All 12 versions



Employee	Publication	Citation
Lipyeow	XPathLearner ...	84
Lipyeow	Characterizing...	38
Haixun	Clustering by ...	308
Haixun	Mining concept ...	424
...

The Simple Solution

Loop

Q = set of keyword queries

Foreach q in Q

Send q to Google Scholar

Scrape the first few pages into tuples

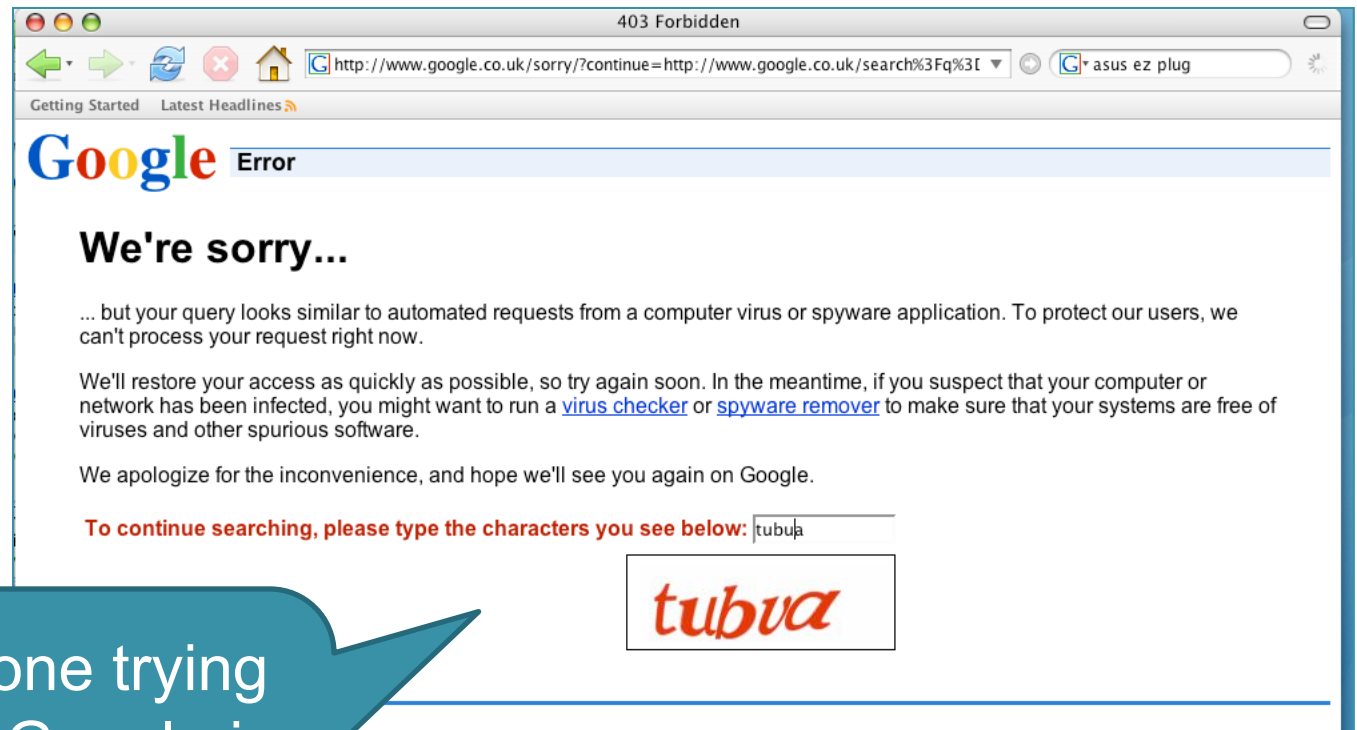
Update local relation using scraped tuples

Sleep for t seconds

End Loop

- Query Google Scholar using researcher's name and/or publication title to get
 - new publications and
 - updated citation counts

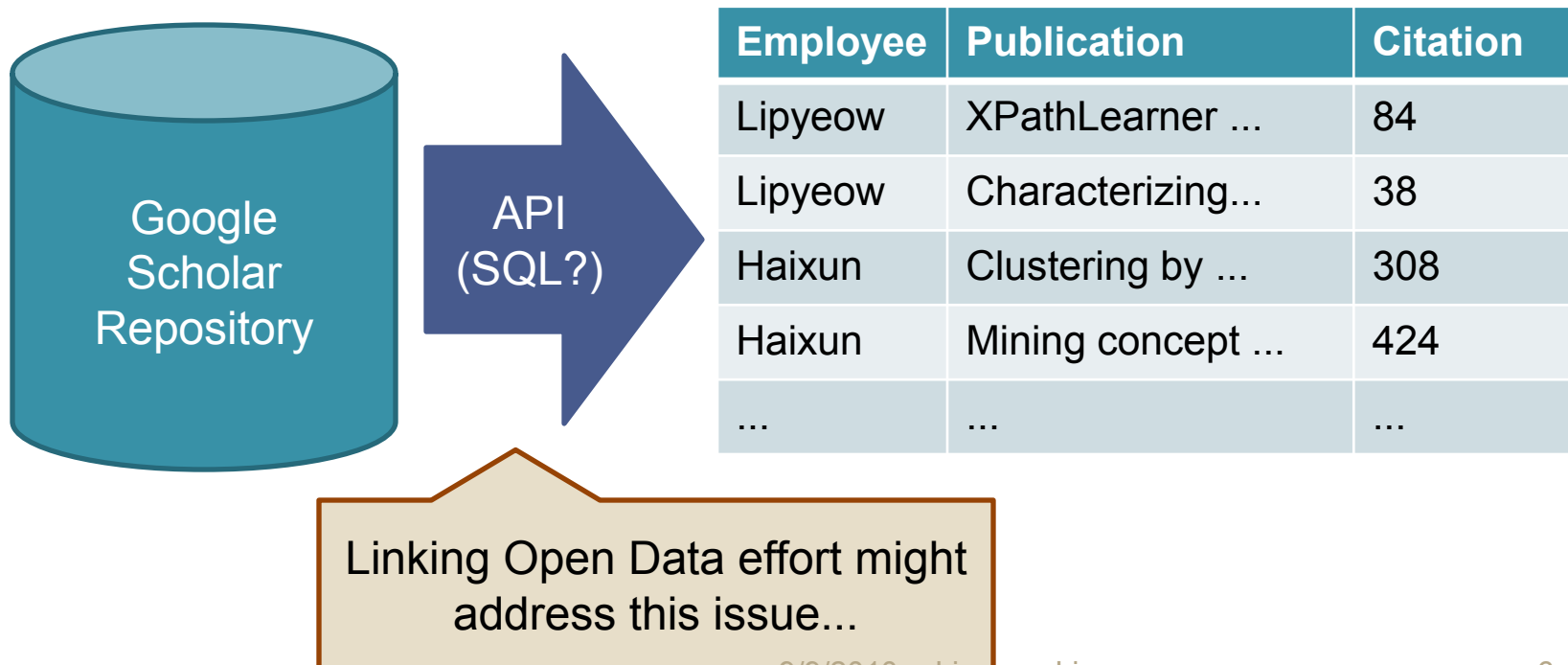
Problem with the Simple Solution



Everyone trying
to use Google in
the building got
this screen !

The Elegant Solution

- All this hacking (including the solution I am about to present) could be avoided if there was an API to get structured relations from Google Scholar.



But ...

- Such API's don't exist (yet?)
- And ...

I need those
citation counts
by next week!



Problem Statement

- Local database periodically synchronizes its data subset with the data source
- Data source supports keyword query API only
- Extract relations from the top k results (ie first few result pages) to update local database

At each synchronization,
find a set of queries that will maximize the
“content freshness” of the local database.

- Only relevant keywords are used in the queries
- Keywords cover the local relation
- Number of queries should be minimized
- Result size should be minimized

NP-Hard by
reduction to
Set Cover

Picking the Right Queries ...

Loop

Q = set of keyword queries

Foreach q in Q

Send q to Google Scholar

Scrape the first few pages into tuples

Update local relation using scraped tuples

Sleep for t seconds

End Loop

- The simple algorithm is fine, we just need to pick the right queries...
 - Not all tuples are equal – some don't get updated at all, some are updated all the time
 - Some updates are too small to be significant

Greedy Probes Algorithm

1. **Q** = empty set of queries
2. NotCovered = set L of local tuples
3. While not **stopping condition** do
4. K = Find all keywords associated with NotCovered
5. Pick **q** from PowerSet(K) using **heuristic equation**
6. Add **q** to **Q**
7. Remove tuples associated with q from NotCovered
8. End While

Could be based on size of Q or coverage of L

- What should greedy heuristic do ?
 - **Local coverage** : a good query will get results to update as much of the local relation as possible
 - **Server coverage** : a good query should retrieve as few results from the server as possible.
 - A good query updates the **most critical** portion of the local relation to maximize “**content freshness**”

Content Freshness

Server

Employee	Publication	Citation
Lipyeow	XPathLearner	87

Local

Employee	Publication	Citation
Lipyeow	XPathLearner	84

- Weighted tuple dissimilarity
 - Some tuples are more important to update
 - $= w(local) * d(local, server)$
- Content Freshness

$$D(L, S) = \frac{1}{|L|} \sum_{l \in L} w(l) \cdot d(l, s)$$

- Example:
 - $w(l) = l.citation = 84$
 - $d(l, s) = |l.citation - s.citation| = 3$

Catch: local DB does not know the current value of citation on the server!

Content Freshness (take 2)

- Estimate the server value of citation using an update model based on
 - Current local value of citation
 - Volatility of the particular citation field
 - Time elapsed since last sync.

$$D(L, S) = \frac{1}{|L|} \sum_{l \in L} w(l) \cdot F(l, t)$$

$F(l, t)$ estimates the dissimilarity between the local tuple and the server tuple at time t assuming an update model

Greedy Heuristic

- Query efficiency

$$q = \arg \max_{q \in P(K)} \frac{|LocalCoverage(q)|}{|ServerCoverage(q)|}$$

- To give higher priority to “unfresh” tuples, we weight the local coverage with the freshness

$$q = \arg \max_{q \in P(K)} \frac{\sum_{l \in LocalCoverage(q)} w(l) \cdot F(l, t)}{|ServerCoverage(q)|}$$

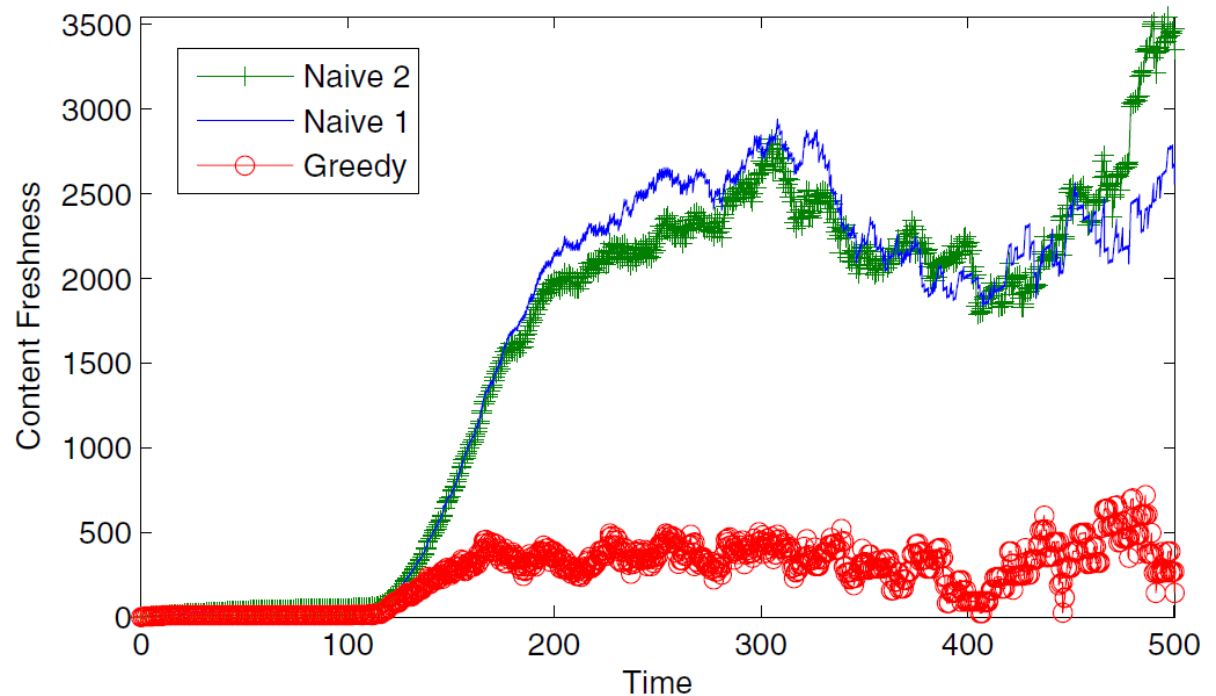
- Catch: local DB does not know server coverage!
 - Estimate server coverage using statistical methods
 - Estimate server coverage using another sample data source (eg. DBLP)

Experiments

- Data sets:
 - Synthetic data
 - Paper citations (this presentation)
 - DVD online store
- Approximate Powerset(K) with all keyword pairs
- Result Extraction
 - Method 1: scan through all result pages
 - Method 2: scan only the first result page

Content Freshness

- Synthetic citation data based on known statistics
- A Poisson-based update model used to estimate freshness
- 10 queries are sent at each sync
- Naive 1 & 2 sends simple ID-based queries



Optimizations

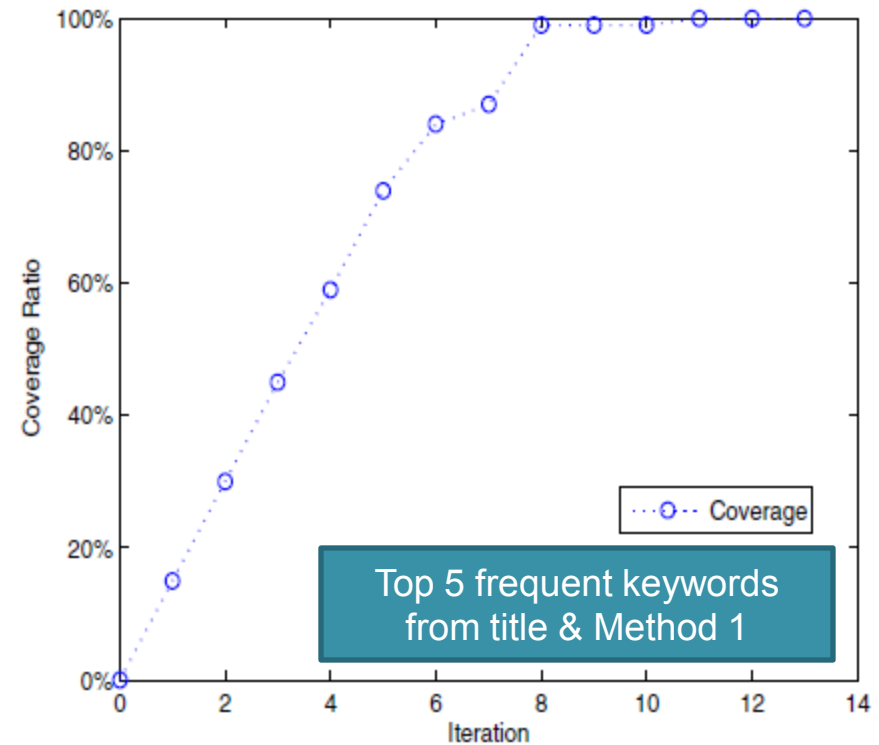
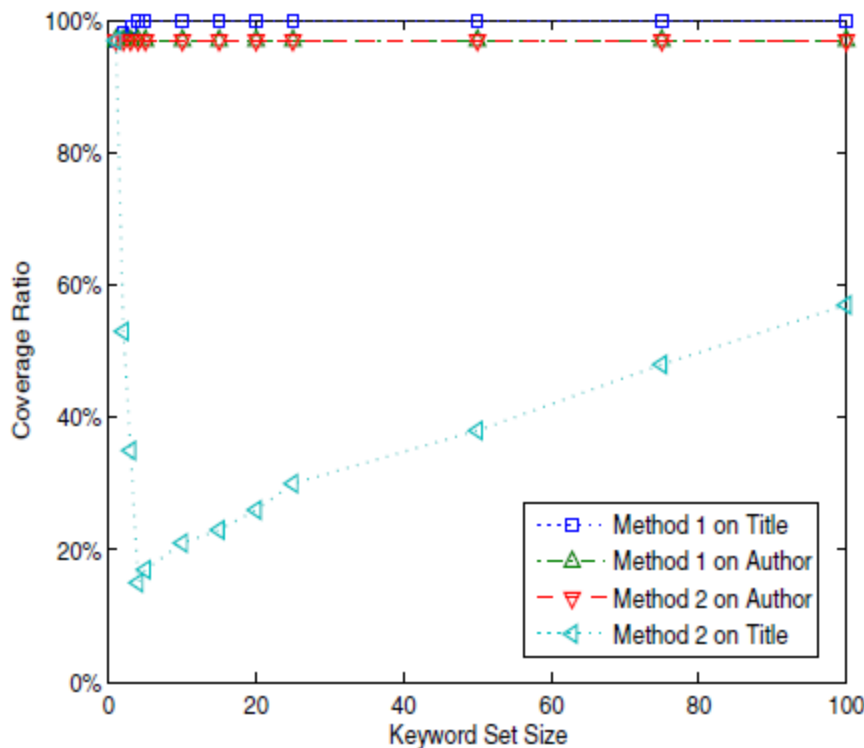
1. Q = empty set of queries
2. NotCovered = set L of local tuples
3. While not **stopping condition** do
4. K = Find all keywords associated with NotCovered
5. Pick q from PowerSet(K) using **heuristic equation**
6. Add q to Q
7. Remove tuples associated with q from NotCovered
8. End While

K can be large

Power Set is exponential

- Approximate K using most frequent k keywords
- Approximate Power Set using subsets up to size $m=2$ or 3 .

Coverage Ratio



- Coverage ratio is the fraction of local tuples covered by a set of queries
- Result Extraction
 - Method 1: scan through all result pages
 - Method 2: scan only the first result page

Conclusion

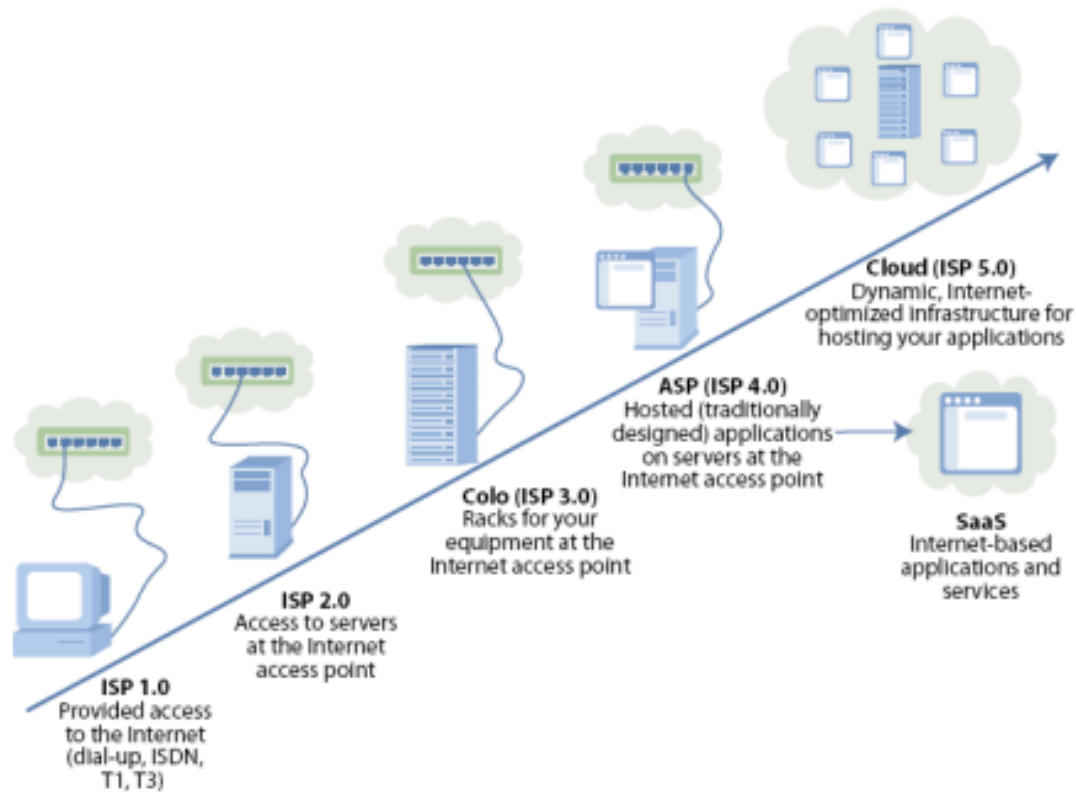
- Introduced the problem of maintaining a local relation extracted from a web source via keyword queries
- Problem is NP-Hard, so design a greedy heuristic-based algorithm
- Tried one heuristic, results show some potential
- Still room for more work – journal paper



Cloud-based Parallel DBMSs

Cloud Computing

Figure 3 Cloud Computing: The Latest Evolution Of Hosting



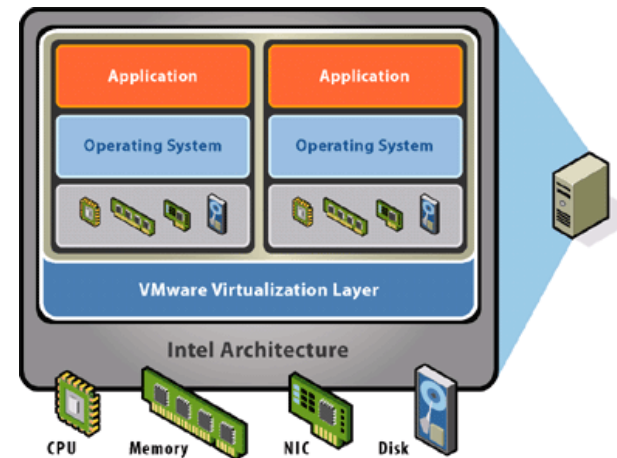
44229

Source: Forrester Research, Inc.

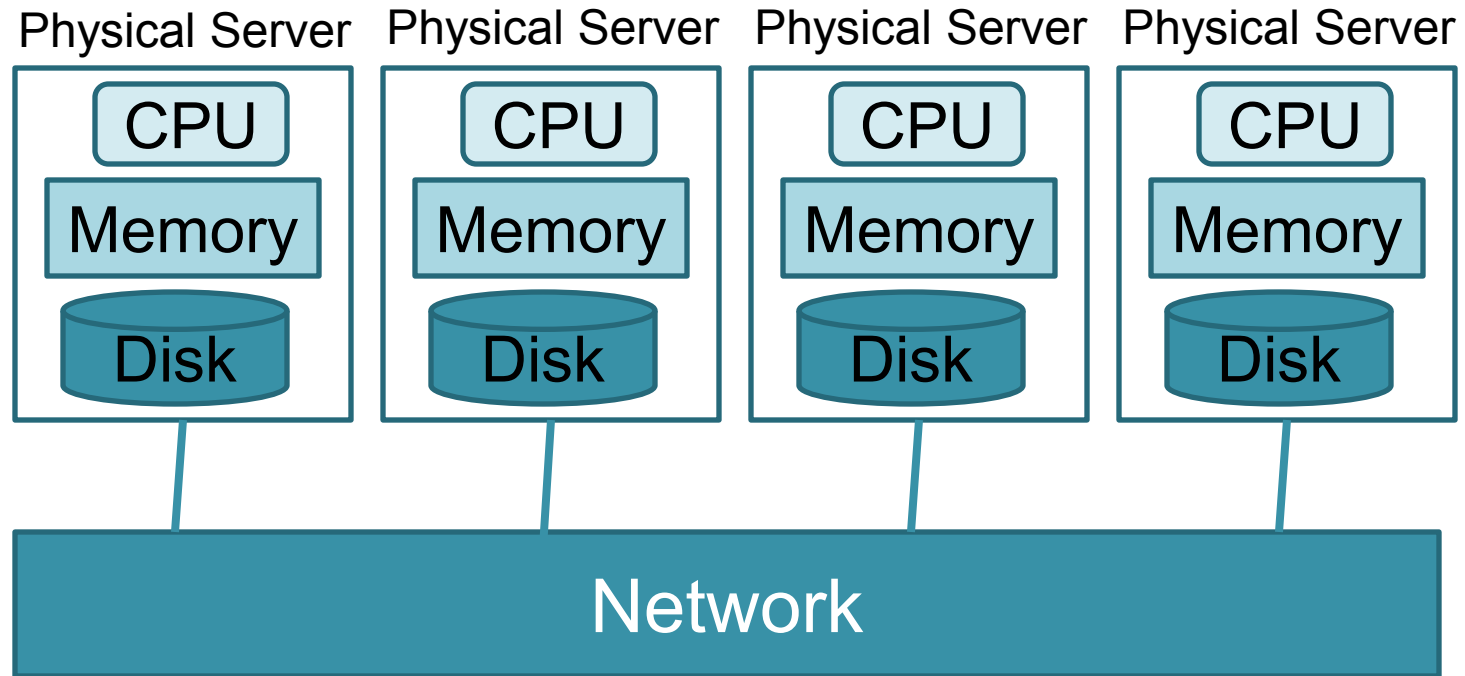
Highlights of the Cloud Computing Landscape



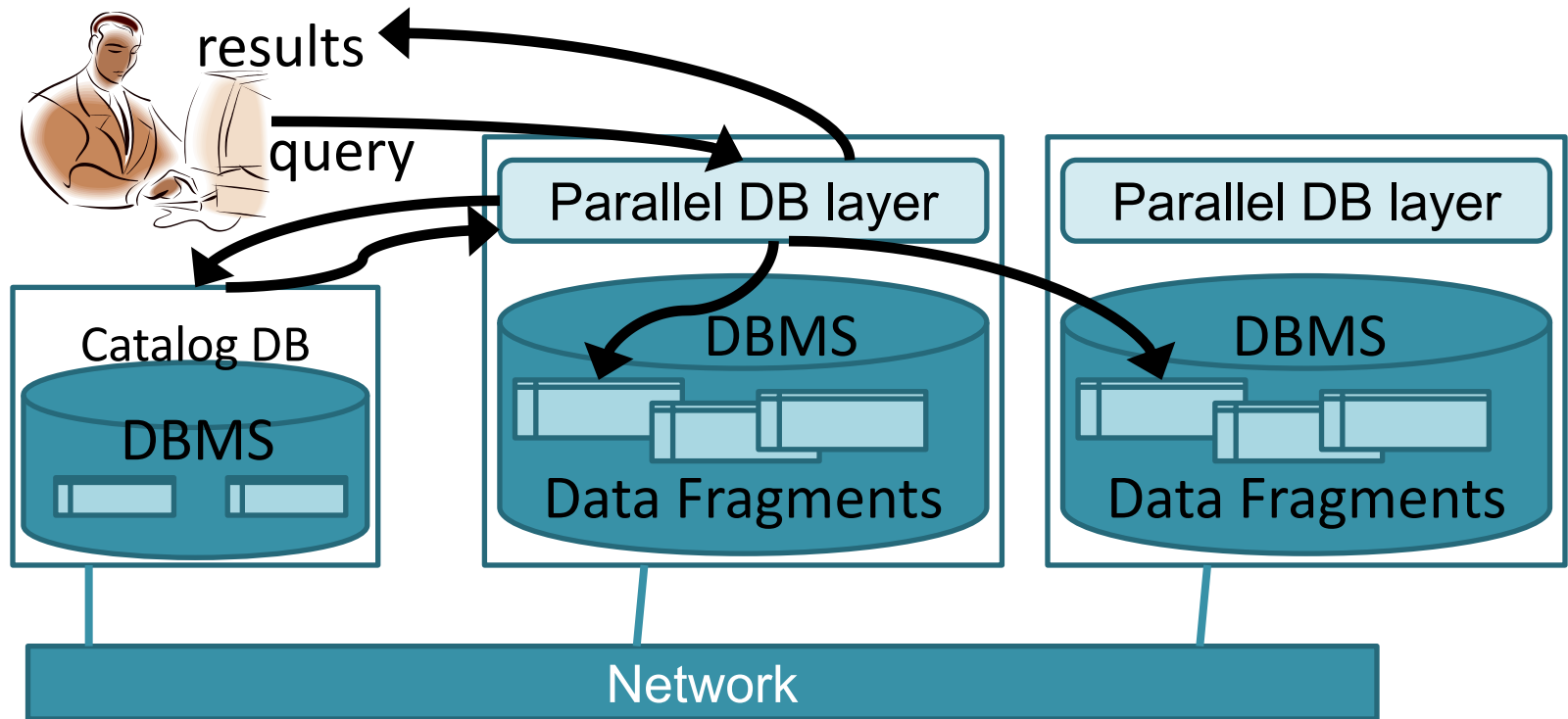
From <http://blogs.zdnet.com/Hinchcliffe>



Shared-Nothing Parallel Database Architecture



Logical Parallel DBMS Architecture



Horizontal Fragmentation: Range Partition

sid	sname	rating	age
22	dustin	7	45
29	brutus	1	33
31	lubber	8	55
32	andy	4	23
58	rusty	10	35
64	horatio	7	35

Range Partition on rating

- Partition 1: $0 \leq \text{rating} < 5$
- Partition 2: $5 \leq \text{rating} \leq 10$

```
SELECT *  
FROM Sailors S
```

```
SELECT *  
FROM Sailors S  
WHERE age > 30
```

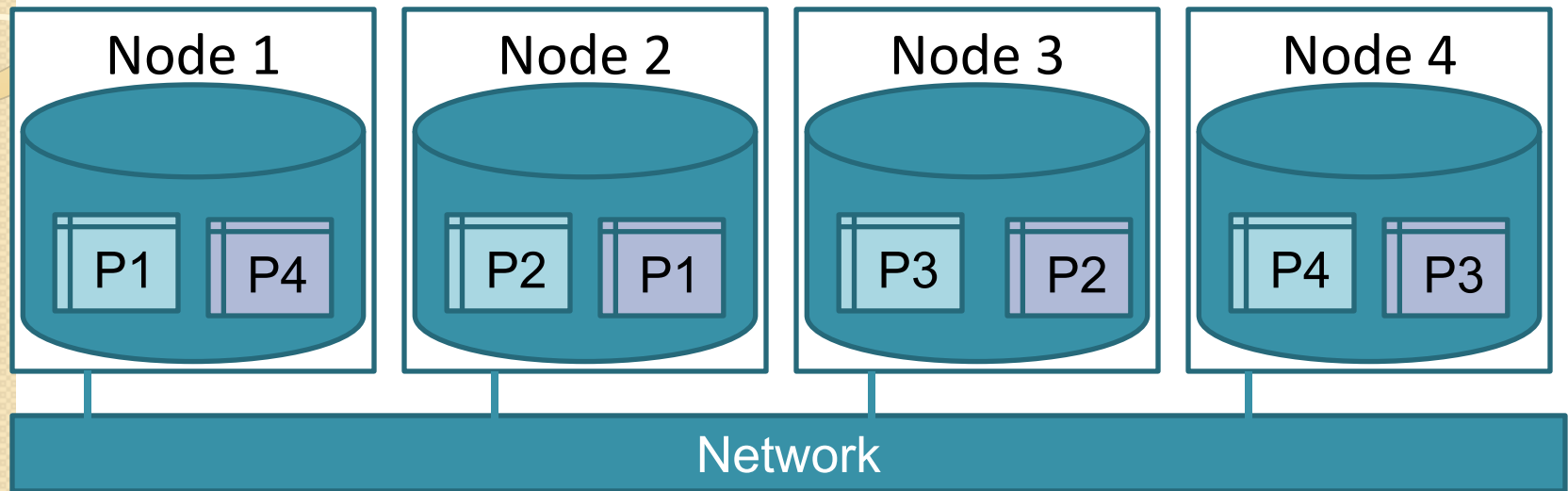
Partition 1

sid	sname	rating	age
29	brutus	1	33
32	andy	4	23

Partition 2

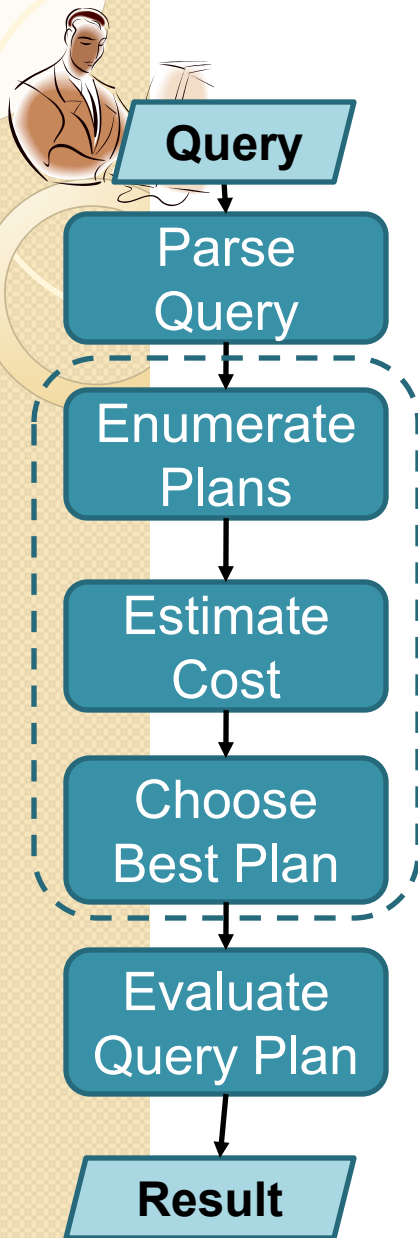
sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35
64	horatio	7	35

Fragmentation & Replication



- Suppose a table is fragmented into 4 partitions on 4 nodes
- Replication stores another partition on each node

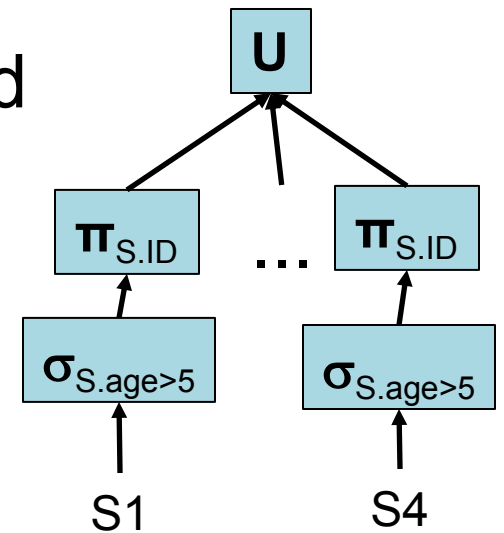
Query Optimization



SELECT S.ID
FROM Sailors S
WHERE age > 30

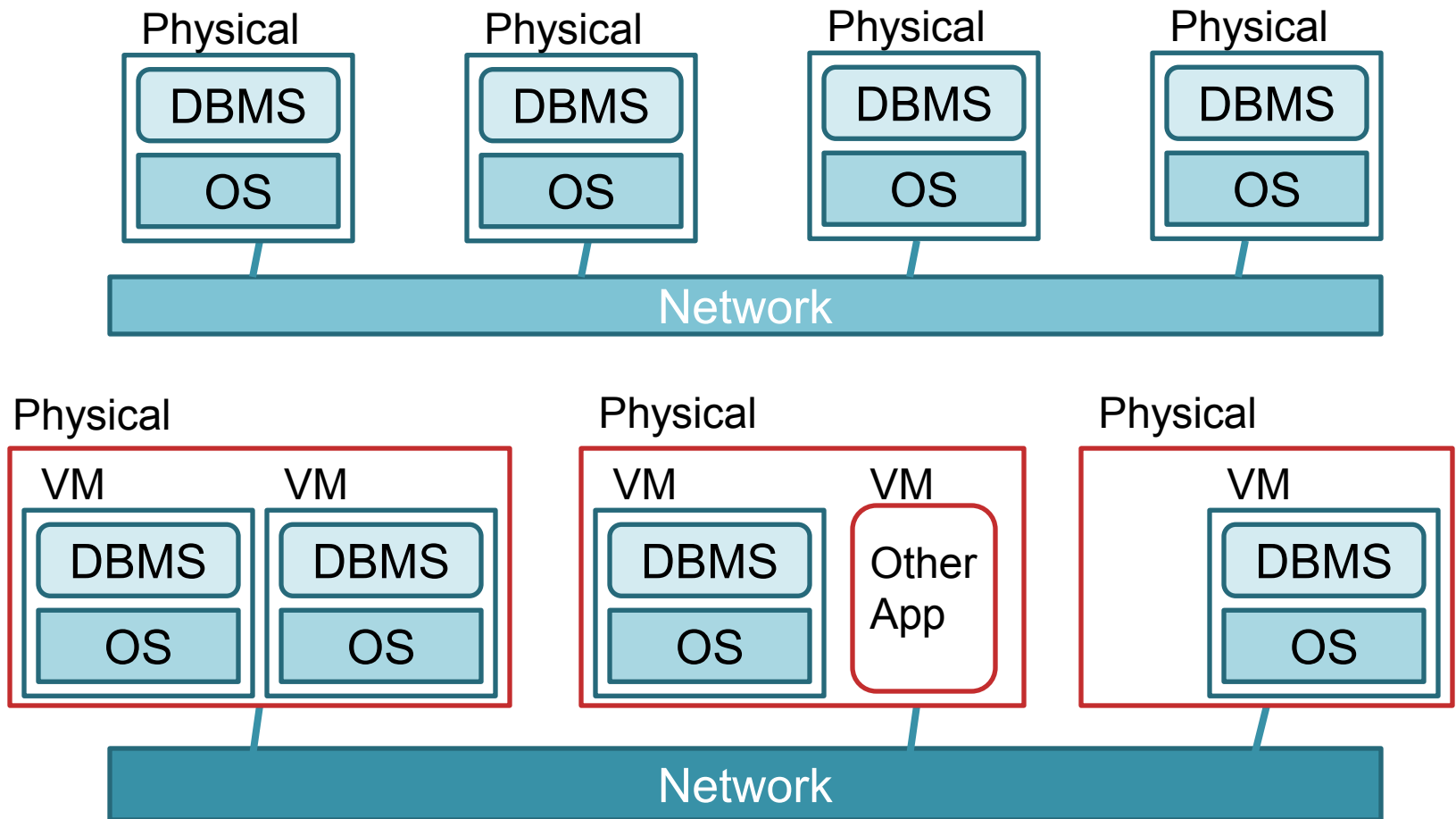
$$\begin{aligned} & \pi_{ID}(\sigma_{age>30}(S)) \\ & \equiv \pi_{ID}(\sigma_{age>30}(S1 \cup S2 \cup S3 \cup S4)) \\ & \equiv \bigcup_{i=1..4} (\pi_{ID}(\sigma_{age>30}(S_i))) \end{aligned}$$

- Sailors fragmented and replicated on 4 nodes
 - S1, S2, S3, S4
 - S1r, S2r, S3r, S4r
- Estimate cost
 - Size of temporary results
 - CPU processing cost
 - Disk IO cost
 - Shipping temp. results



Choice of
S4 or S4r

What changed in the cloud ?



- Virtualization “messes up” CPU, IO, network costs
- Migration of VMs possible in principle

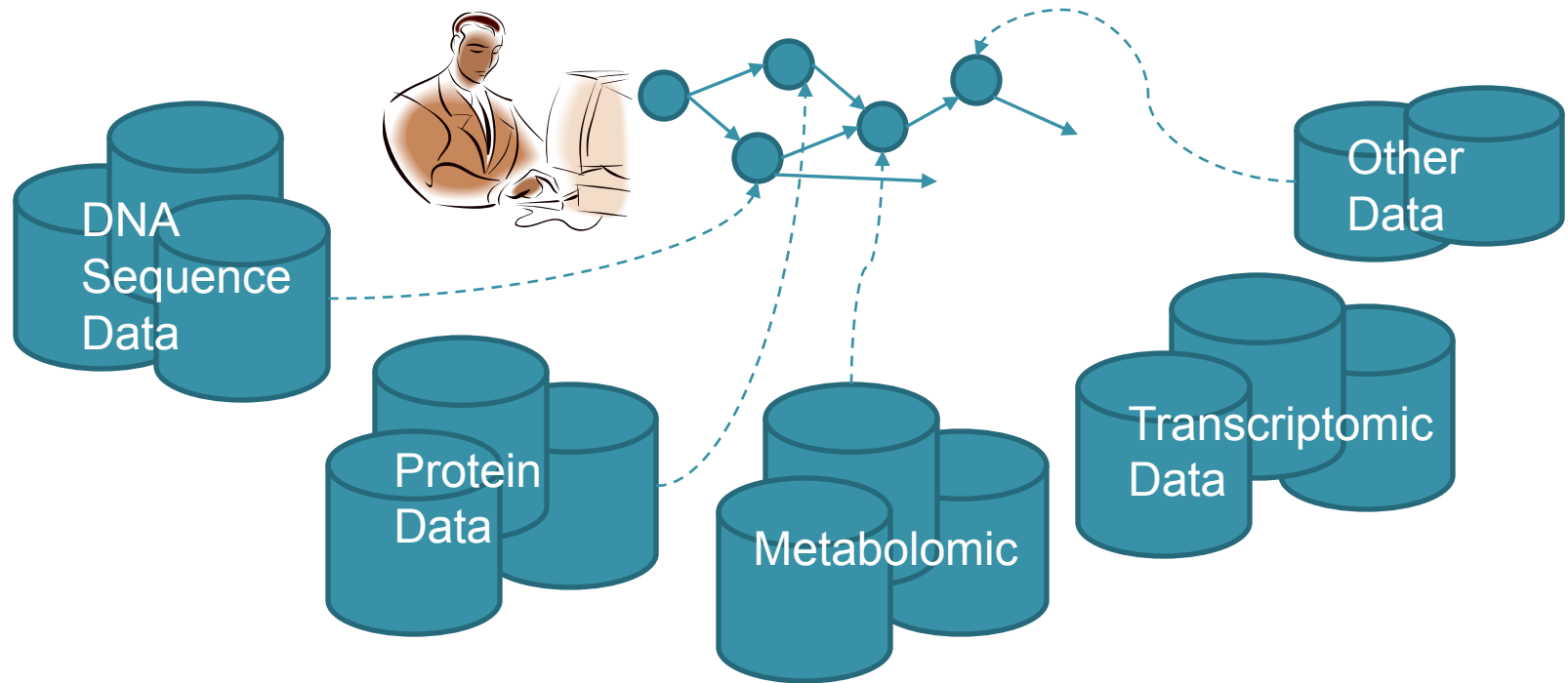
Problems & Tasks

- **Query optimization**
 - What is the impact of virtualization on cost estimation?
 - What new types of statistics are needed and how do we collect them ?
 - CPU cost
 - Disk I/O cost
 - Network cost
- **Enabling elasticity**
 - How can we organize data to enable elasticity in a parallel DBMS ?
- **Scientific applications** (eg. astronomy)
 - Semantic rewriting of complex predicates



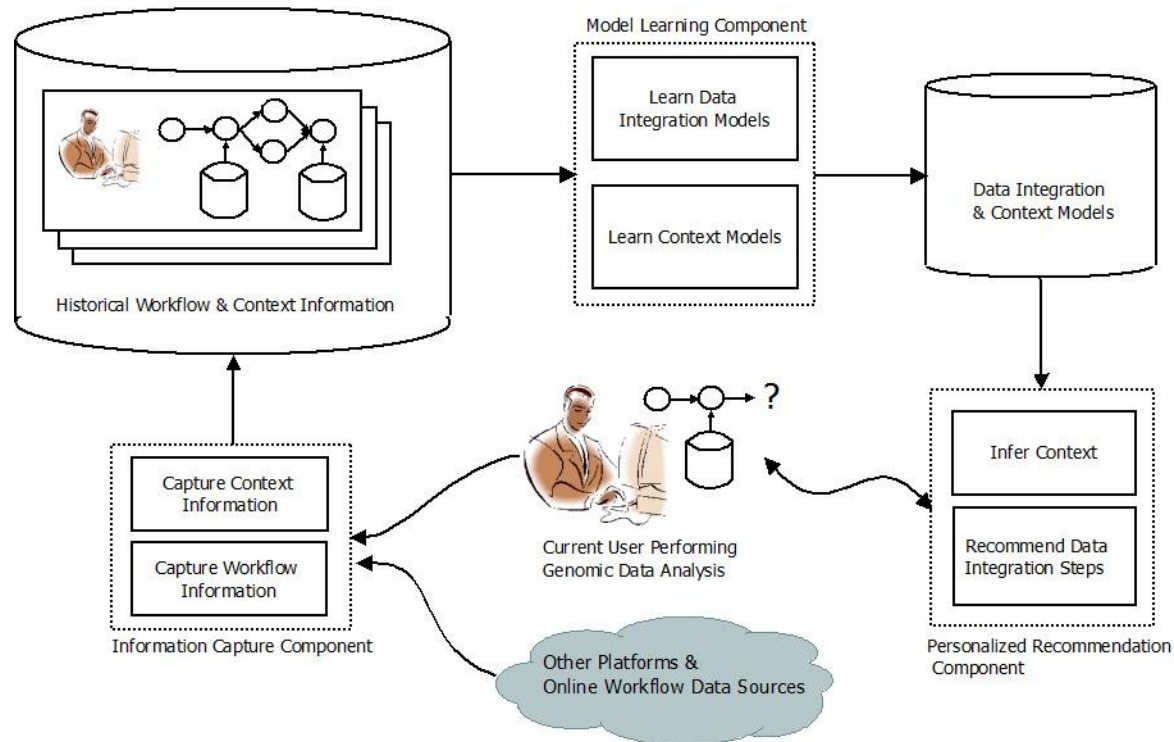
Mining Workflows for Data Integration Patterns

Bio-Informatics Scenario



- Each category has many online data sources
- Each data source may have multiple API and data formats
- Workflow is like a program or a script
 - A connected graph of operations

A Data Integration Recommender



- **Data integration patterns**
 - Generalize on key-foreign key relationships
 - Associations between schema elements of data and/or processes
- Analyze **historical workflows** to extract data integration patterns
- Make personalized recommendations to users as they create workflows

Problems & Tasks

- What are the different types of data integration patterns we can extract from workflows ?
- How do we model these patterns ?
- How do we mine workflows for these patterns ?
- How do we model context ?
- How do we make recommendations ?



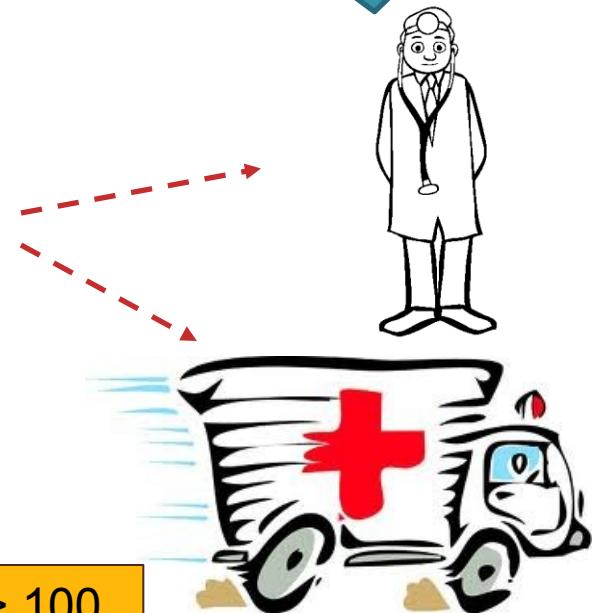
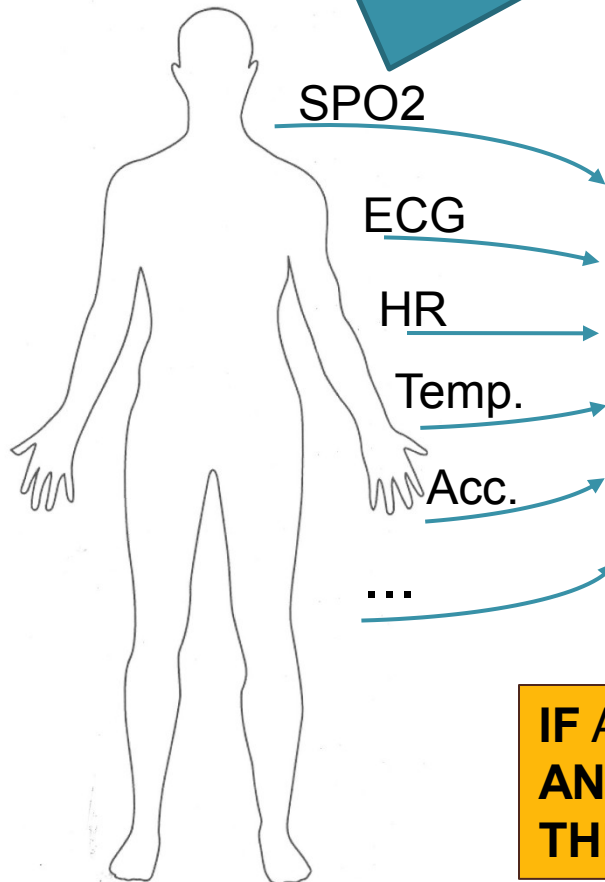
Energy Efficient Complex Event Processing

Telehealth Scenario

Wearable sensors transmit
vitals to cell phone via
wireless (eg. bluetooth)

Phone runs a complex
event processing (CEP)
engine with rules for
alerts

Alerts can
notify
emergency
services or
caregiver



IF Avg(Window(HR)) > 100
AND Avg(Window(Acc)) < 2
THEN SMS(doctor)

Energy Efficiency



- Energy consumption of processing
 - **Sensors**: transmission of sensor data to CEP engine
 - **Phone**: acquisition of sensor data
 - **Phone**: processing of queries at CEP engine
- Optimization objectives
 - Minimize energy consumption at phone
 - Maximize operational lifetime of the system.
- Ideas:
 - Batching of sensor data transmission
 - Moving towards a pull model
 - Order of predicate evaluation

Evaluation Order Example

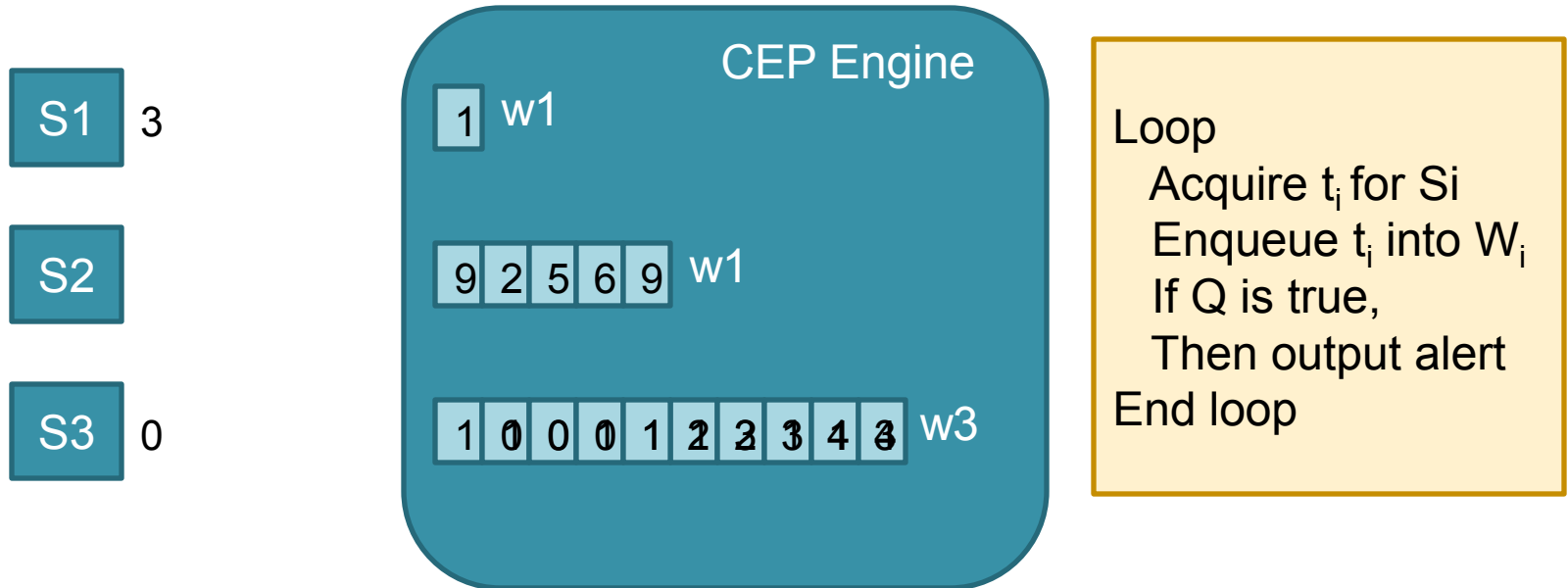
if $\text{Avg}(S2, 5) > 20$ AND $S1 < 10$ AND $\text{Max}(S3, 10) < 4$ then email(doctor).

Predicate	$\text{Avg}(S2, 5) > 20$	$S1 < 10$	$\text{Max}(S3, 10) < 4$
Acquisition	$5 * .02 = 0.1 \text{ nJ}$	0.2 nJ	$10 * .01 = 0.1 \text{ nJ}$
Pr(false)	0.95	0.5	0.8
Acq./Pr(f)	0.1/0.95	0.2/0.5	0.1/0.8

- Evaluate predicates with lowest energy consumption first
- Evaluate predicates with highest false probability first
- Hence, evaluate predicate with lowest normalized acquisition cost first.

Continuous Evaluation

if $\text{Avg}(S2, 5) > 20$ AND $S1 < 10$ AND $\text{Max}(S3, 10) < 4$ then email(doctor).



- Push model
 - Data arrival triggers evaluation
- Pull model:
 - Engine decides when to perform evaluation and what order.
 - Rate problem

Problems and Tasks

- What are the energy cost characteristics of different query evaluation approaches with different alert guarantees ?
- What is the impact of batching and pull-based transmission ?
- Design novel energy efficient evaluation algorithms
- How do we estimate the probability of true/false of predicates ?
- Experiment on simulated environment
- Experiment on Android phone & shimmer sensor environment

Agenda

1. **SIGMOD 2010 Paper**: Optimizing Content Freshness of Relations Extracted From the Web Using Keyword Search
2. **Cloud-based Parallel DBMS**
3. **Mining Workflows for Data Integration Patterns**
4. **Energy Efficient Complex Event Processing**