RIPPLE: A SCALABLE FRAMEWORK FOR DISTRIBUTED PROCESSING OF RANK QUERIES

George Tsatsanifos (NTUA)
Dimitris Sacharidis (“Athena” RC)
Timos Sellis (RMIT)

17th International Conference on Extending Database Technology
March 26, 2014 - Athens, Greece
Outline

1 Introduction

2 Related Work
   • P2P Overlay Networks
   • Distributed Schemes for Rank Queries

3 Query Processing in RIPPLE
   • Framework
   • Top-\(k\) & Skyline Queries
   • Search Result Diversification

4 Experiments
   • Evaluation of Skyline Queries
   • Evaluation of Search Result Diversification

5 Conclusions
Why Rank Queries?

- For users with an “unspecified” interest in certain criteria,
  - cheapest, most comfortable accommodation
  - low fare tickets to the most exotic destination.

- Query type with unique characteristics
  - Different than conventional query types, e.g. range queries.
  - Optimization-like search methods employed.
  - Unbounded search area (generally).

- They are hard to compute
  - and even harder in a distributed setting
  - consume significant resources.
Outline

1. Introduction

2. Related Work
   - P2P Overlay Networks
   - Distributed Schemes for Rank Queries

3. Query Processing in RIPPLE
   - Framework
   - Top-k & Skyline Queries
   - Search Result Diversification

4. Experiments
   - Evaluation of Skyline Queries
   - Evaluation of Search Result Diversification

5. Conclusions
Distributed Hash-Tables (DHTs)

- Provision of a lookup service.
- Consistent hashing assigns ownership of a \((\text{key}, \text{val})\) pair.
- A typical interface example:
  - \text{join}()
  - \text{depart}()
  - \text{insert}()
  - \text{lookup}()
Content **Addressable Network (CAN)**

- Keys are hashed into $d$ dimensional space with $n$ zones.
- Two nodes are neighbors if they overlap along $d - 1$ dims.
- Routing based on choosing the node nearest to destination.
P2P Overlay Networks

**BATON**

- A balanced, distributed, binary tree.
- Nodes linked to nodes on the same level, parents, children.
- Nodes must have full routing tables before having children.
- Intervals assigned according to an inorder traversal.
MIDAS under the scope

**Multi-Attribute Indexing for Distributed Architecture Systems**

- Leaf nodes correspond to actual peers
- Internal nodes serve as routing directives.
- Tree hierarchy represented by *split history* and *split points*.
- A peer knows another peer from each subtree it does not belong to, for each node in its path to the root.
Outline

1. Introduction
2. Related Work
   - P2P Overlay Networks
   - Distributed Schemes for Rank Queries
3. Query Processing in RIPPLE
   - Framework
   - Top-k & Skyline Queries
   - Search Result Diversification
4. Experiments
   - Evaluation of Skyline Queries
   - Evaluation of Search Result Diversification
5. Conclusions

RIPPLE: A Scalable Framework for Distributed Processing of Rank Queries
Introduction

Related Work

Query Processing in RIPPLE

Experiments

Conclusions

Distributed Schemes for Rank Queries

Top-$k$ Queries for Distributed Settings

Efforts like TPUT and KLEE extend TA.

1. Fetch the $k$ best entries from each of the $m$ cohort nodes.
2. Compute the partial score of the $k$-th best entry $\tau$.
3. Ask for records with score greater than $\frac{\tau}{m}$.
4. Eliminate entries with optimistic score less than $\tau'$.
5. Fetch missing scores for remaining candidates.
   - Cohort nodes perform random accesses.
Distributed Schemes for Rank Queries

Top-$k$ Queries for Horizontally Distributed Data

**SPEERTO**

- Each node maintains a subset of the relation.
- Nodes compute their $k$-skybands.
  - sets of points dominated by $k - 1$ points at most.
- Local results are then aggregated by super-peers.

![Diagram of SPEERTO algorithm](image)
Distributed SkyLines (DSL)

- Operates on top of CAN.
- Formed query plan corresponds to an ad-hoc multicast tree.
- The hierarchy is built such that only nodes that cannot dominate on each other are queried in parallel.
- Partial ordering captures the computational dependencies.
- Mutually independent from a skyline perspective at level $i$.
  - Therefore, their calculations can proceed in parallel.
# Distributed SkyLines (DSL)

- The query is propagated along the edges of the hierarchy.
- Nodes perform local computation.
- Aggregate and propagate the results back on same path.
  - Ineligible data are discarded.

![Diagram of Distributed SkyLines (DSL)](image)
Skyline Space Partitioning (SSP)

Incorporates a z-curve over BATON, a balanced binary tree.

1. Node responsible for origin of axes computes local skyline guaranteed to be in the global skyline set.

2. Selects the most dominating point in an effort to
   - to refine the search space
   - to exclude from search a significant part of the overlay

3. Query is forwarded recursively to all known qualified nodes.

4. Issuer congregates all partial results and discards false hits.
Skyframe

Essentially constitutes a generalization of SSP.

1. The issuer node serves as a coordinator.
2. Queries nodes responsible for a min value in any dimension.
3. Coordinator processes the intermediate results.
4. Determines the next group of nodes to be queried (if at all).
Outline

1. Introduction
2. Related Work
   - P2P Overlay Networks
   - Distributed Schemes for Rank Queries
3. Query Processing in RIPPLE
   - Framework
   - Top-k & Skyline Queries
   - Search Result Diversification
4. Experiments
   - Evaluation of Skyline Queries
   - Evaluation of Search Result Diversification
5. Conclusions
RIPPLE: A Scalable Framework for Distributed Processing of Rank Queries

Framework

Intuition

- RIPPLE combines two different approaches.
  - Operates on two levels.
- Compromises *latency* with *congestion*.
- Query forwarded toward the most promising overlay parts.
- Resolved fast within clusters of highly ranked nodes.
  - using a rank-aware shower-like method.
- Whilst suboptimal neighborhoods are effectively pruned.
- Generalized approach for supporting different query types.
How and Why? -i-

- We already know why a tuple is better than another... but how can an overlay node outrank another?
- Is it because it contains better tuples? Just partly!
- Keep in mind the two principal tasks of a node:
  1. Answer queries concerning local data.
  2. Forward network traffic to more relevant nodes.

(a) k-d tree  (b) u’s links  (c) u’s view  (d) actual zone
How and Why?

- Thereby, a node can be seen as a *mediator* responsible for a whole *region* which encompasses its own *zone*.
- A region depends on the viewpoint of the node it serves.
- A better node provides access to better data (local or not)
- An overlay node is better than another,
  - if its region might contain a better ranked tuple.
  - if its region is closer to the origin of axes for skyline queries.

(a) k-d tree  (b) u’s links  (c) u’s view  (d) actual zone
Outline

1. Introduction
2. Related Work
   - P2P Overlay Networks
   - Distributed Schemes for Rank Queries
3. Query Processing in RIPPLE
   - Framework
   - Top-k & Skyline Queries
   - Search Result Diversification
4. Experiments
   - Evaluation of Skyline Queries
   - Evaluation of Search Result Diversification
5. Conclusions
Problem Formulation

Problem Definition

Find the $k$ best tuples according to a unimodal function.

Specifications

- User-defined increasingly monotone aggregate function $f$.
  - Such as the weighted sum over all dimensions.
- Increasingly monotone: $\forall i, p_i \leq p_i' \Rightarrow f(p_i) \leq f(p_i')$. 

RIPPLE: A Scalable Framework for Distributed Processing of Rank Queries
Top-$k$ & Skyline Queries

Processing Top-$k$ Queries

Query state $⟨\bar{w}, k, m, \tau, r⟩$

1. Append the best $k - m$ tuples for $m < k$.
2. Also, insert the next (up to $m$) best tuples with score greater than $\tau$, for they outperform the $m$-th best entry.
3. Decrement $r$ and forward the updated request:
   - $r > 0$ Prioritize links in terms of their potential and forward the request sequentially to the next in the list that can still improve the answer.
   - $r \leq 0$ Query spans all known nodes whose region might contain better ranked items.
The Pareto Optimum / Maximum Vector Problem

Definition

- Given $d$-dimensional feature space $D$ and set of objects $O$.
- A point $p \in O$ dominates a point $p'$, iff $\forall i, p_i \leq p'_i$ and $\exists j$ such that $p_j \in p'_j$.
- $SKY \subseteq O$ the set of points not dominated by any other.
- Top-1 object for any monotone function belongs to $SKY$. 
Processing Principles

Retrieve all tuples with no better tuple in all dimensions.

Methodology

1. Exploit the additivity of the skyline operator.
   - Assume some partitioning of dataset $S = \bigcup_i S_i$ into nodes.
   - Then, $SKY(S) = SKY(\bigcup_i S_i) = SKY(\bigcup_i SKY(S_i))$.

2. Prune sub-optimal local data objects through filtering.
   - Use current result to discard dominated local points.
   - Or prevent from reading them in the first place.
   - Reduces traffic as only local points transferred.

3. Peer pruning based on local routing information.
   - Use local/intermediate results to exclude overlay parts.
   - Check whether can contribute from its responsibility area.
   - Exploits the dominance relation.
Skyline Computation

Upon reception of a query along with an intermediate result,

1. Retrieve all local points not dominated by current result.
2. Combine them to form a new set in such a way that no point dominates on another.
3. False hits are discarded, and thus, net traffic is minimized.
4. Forward updated query to neighbors that can contribute.
   - If $r > 0$, the next known node closest to $\vec{0}$ is selected.
   - Otherwise, all non-dominated links receive the query.
5. Eventually, issuer merges all local and intermediate results.
Overlay Structure Revisited

- Recipients must be part of the answer more often than not.
- Highly relevant nodes responsible for non-dominated areas
  - They are thus located close to the borders of the keyspace.
- These nodes have specific identifiers for a given setting.
- For $D$ dims as splits alternate
  $$p_0 = (X0 \cdots 0) X0 \{0, D-1\},$$
  $$p_1 = (0X \cdots 0) 0X0 \{0, D-2\},$$
  and so on.
- One pattern per split-dimension with $X$s at the positions that correspond to the splits along this dimension.
- If interested in max values instead, switch zeros for ones.
Overlay Structure Revisited

- Network structure is formed in such a way that if there is at least one such peer within the sibling subtree at level \( j \), a connection is established.
- The \( j \)-th link is chosen so as to comply with any pattern.
- For two-dimensions where the split-dimensions interchange: 
  \[
  p_h = (X0)^*X \text{? and } p_v = (0X)^*0 \text{?}
  \]
Outline

1 Introduction

2 Related Work
   • P2P Overlay Networks
   • Distributed Schemes for Rank Queries

3 Query Processing in RIPPLE
   • Framework
   • Top-k & Skyline Queries
   • Search Result Diversification

4 Experiments
   • Evaluation of Skyline Queries
   • Evaluation of Search Result Diversification

5 Conclusions
Motivation

Counteracting overspecialization

- Results with no (near) duplicates are more preferable.
- Retrieve results that consist of relevant tuples that differentiate from each other.
- Content-based approach as entries selected progressively according to the relevance of the ones that come before it.
- The result is then further refined iteratively by carefully selecting the entries to be replaced and their replacements.
Preliminaries

Compromising contradictory notions

- *Relevance* corresponds to the similarity to the query.
  - e.g., distance from a query point.
- *Diversity* defined by how much the elements in the result differentiate from each other.
- Their convex combination expressed in functions of $S$ and $q$

\[
\begin{align*}
\lambda &\sum_{s \in S} d(s, q) - \frac{1-\lambda}{|S|^2} \sum_{s_1 \in S} \sum_{s_2 \in S} d(s_1, s_2) \\
\lambda &\max_{s \in S} d(s, q) - (1 - \lambda) \min_{s_1 \in S, s_2 \in S} d(s_1, s_2)
\end{align*}
\]
Definitions

Given query object \( q \), a universe of objects \( U \), its subset \( S \subset U \)
- Let ranking function \( f(S|q) \) quantifying \( S \)'s properties.
- Also, let object \( o \in U \setminus S \) to be added in the result.
- Is there a way to capture objects’ eligibility beforehand?
- How is the rank of \( S \) affected by inserting \( o' \) instead?
Definitions

- Let function $\phi(o|S, q) = f(S \cup \{o\}|q) - f(S|q)$, reflecting the change in $S$’s rank and properties.
- It holds for the best object $o’$ to be added $\phi(o’) \leq \phi(o)$.
- For min-sum rank,
  - $\phi(o|S) = \lambda d(o, q) - \frac{1-\lambda}{|S|} \sum_{s \in S} d(o, s)$
  - $\phi([\vec{\ell}, \vec{h}]|\vec{q}, S) = \min_{\vec{p} \in [\vec{\ell}, \vec{h}]} \phi(\vec{p}|\vec{q}, S)$
  - $\phi([\vec{\ell}, \vec{h}]|\vec{q}, S) \geq \lambda \min_{\vec{x} \in [\vec{\ell}, \vec{h}]} d(\vec{x}, \vec{q}) - \frac{1-\lambda}{|S|} \sum_{\vec{s} \in S} \max_{\vec{y} \in [\vec{\ell}, \vec{h}]} d(\vec{y}, \vec{s})$
- We generalize $\phi$ for high-dimensional intervals $[\vec{\ell}, \vec{h}]$ and approximate it with its lower bound.
- Thereby, we are in position of comparing the regions of two nodes in terms of how promising they are.
Search Result Diversification

Refinement

- Starting from an answer-set $S$, how can we improve it?
- We choose $s'$ to be replaced and replacement $o'$ such that
  $$f((S \setminus \{s\}) \cup \{o\}) \leq f((S \setminus \{s\}) \cup \{o\}), \forall s \in S, \forall o \in U \setminus S$$
- Thereby, we would have to examine each element of $S$ and find its replacement $o$, to opt for the best pair.
- But we want to exclude from search the parts of the overlay with suboptimal entries.
- We want to benefit from previous choices, needless to encounter nodes with not as good entries as the last.
In search for a better result

Starting with result $S$, we replace $s$ with $o$ to form $S'$, so that

$$f(S') < f(S)$$

$$f((S \setminus \{s\}) \cup \{o\}) < f(S)$$

$$f(S \setminus \{s\}) + \phi(o \mid S \setminus \{s\}) < f(S \setminus \{s\}) + \phi(s \mid S \setminus \{s\})$$

$$\phi(o \mid S \setminus \{s\}) < \phi(s \mid S \setminus \{s\})$$
Insight

When looking for replacements

- In practice, $\phi$-values correspond to rank thresholds.
- Therefore, $\phi$ serves as a discriminant function.
  - $r > 0$ Best-First Search.
  - $r \leq 0$ Only a subset of nodes accessed in parallel.

\[ \vec{s} = (1, 0), \vec{s}' = (1, 2) \quad (b) \quad \vec{s} = (1, 1), \vec{s}' = (1, 2) \quad (c) \quad \vec{s}' = (-2, 2) \]

Figure: Dividing curves for $\lambda = 0.5$
Sequential Processing

- The next replacement should be better that the previous.
- Not just better than the object it replaces!

\[
f(S'') < f(S')
\]

\[
f(S \setminus \{s_j\} \cup \{o_j\}) < f(S \setminus \{s_i\} \cup \{o_i\})
\]

\[
f(S_j) + \phi(o_j|S_j) < f(S_i) + \phi(o_i|S_i)
\]

\[
\phi(o_j|S_j) < \phi(o_i|S_i) + f(S_i) - f(S_j)
\]

Therefore, replacement’s rank threshold becomes even stricter for \(\delta < 0\). As a result, even less nodes receive the request.
Search Result Diversification

Restricting Search Space

To achieve negative $\delta$-values, the turn we examine $S$’s elements is very important. So, we visit $s_i$ before $s_j$, with $\phi(s_i) \geq \phi(s_j)$.

$$\begin{align*}
 f((S \setminus \{s_i\}) \cup \{s_i\}) &= f((S \setminus \{s_j\}) \cup \{s_j\}) \\
 f(S_i) + \phi(s_i|q,S_i) &= f(S_j) + \phi(s_j|q,S_j) \\
 f(S_i) &\leq f(S_j) \\
 f(S_i) - f(S_j) &\leq 0 \\
 \delta &\leq 0
\end{align*}$$

For $\phi(s_i) \geq \phi(s_j)$, $S_i$ constitutes a better subset than $S_j$. 

RIPPLE: A Scalable Framework for Distributed Processing of Rank Queries
Quintessential Diversification

So, why don’t we try to improve $S_i$ first then?

- Refining $S_j$ becomes more focused since we only search for objects that would make it at least as good as $S_i \cup \{o_i\}$.
- The next replacements must be very highly ranked in order to compensate for starting from a worse partial result $S_j$.
- Only a small part of keyspace corresponds to such quality.
- In effect, whole overlay parts are excluded from search.
- *Despite starting from the best subset, we still select the best pair, neither the first nor the last necessarily.*
### Outline

1. **Introduction**
2. **Related Work**
   - P2P Overlay Networks
   - Distributed Schemes for Rank Queries
3. **Query Processing in RIPPLE**
   - Framework
   - Top-$k$ & Skyline Queries
   - Search Result Diversification
4. **Experiments**
   - Evaluation of Skyline Queries
   - Evaluation of Search Result Diversification
5. **Conclusions**

**RIPPLE: A Scalable Framework for Distributed Processing of Rank Queries**
Setting

**Parameters**

Network size: $2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}, 2^{16}, 2^{17}$

Dimensionality: 2, 3, 4, 5, 6, 7, 8, 9, 10

Relevance/Diversity trade-off: 0, .1, .2, .3, .4, .5, .6, .7, .8, .9, 1

Result-size: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

**Datasets**

*NBA dataset:* six-dimensional of 22K records (1946-2009).

*MIRFLICKR:* five-bucket edge histograms descriptor (MPEG7)

*SYNTH:* a clustered dataset of varied dimensionality.
Skyline Queries Performance
in terms of network size

- Logarithmic performance due to overlay characteristics.
- DSL takes advantage of the increased dimensionality.
Skyline Queries Performance

in terms of dimensionality

DSL

1. for low-dimensional domains is outperformed,
   - for the number of routing alternatives relies on $d$.

2. in high-dimensional spaces performance ameliorates,
   - more links lead to larger neighborhoods
   - mostly relevant nodes accessed
   - increased maintenance costs.

(c) latency  
(d) congestion
Diversified Search Performance

in terms of relevance/diversity trade-off

- The number of iterations required plays a dominant role.
- Search becomes very limited for $\lambda$ values close to 0 or 1.
- Search directed towards the nodes either very close to the query, or very distant, along the borders.

(e) latency

(f) congestion
Outline

1 Introduction

2 Related Work
   - P2P Overlay Networks
   - Distributed Schemes for Rank Queries

3 Query Processing in RIPPLE
   - Framework
   - Top-k & Skyline Queries
   - Search Result Diversification

4 Experiments
   - Evaluation of Skyline Queries
   - Evaluation of Search Result Diversification

5 Conclusions
Conclusions

All in all,

- we have addressed the three main types of rank queries,
  1. Top-$k$ queries
  2. Skyline queries
  3. Result diversification

- we investigated trade-offs between latency and congestion.
- we combined best-first search with fast multicast schemes.
- a supplementary structure for resolving skyline queries.
Questions?
Insight

- For $\lambda \neq 0.5$ balance between relevance and diversity breaks.
- We are interested in the area outside the curves for $\lambda < 0.5$.
- Only nodes who overlap with this area are accessed.

Figure: Dividing curves for $\lambda = 0.3$

(g) $\vec{s} = (1, 0)$, $\vec{p} = (1, 2)$
(h) $\vec{s} = (1, 1)$, $\vec{p} = (1, 2)$
(i) $\vec{s} = (1, 1)$, $\vec{p} = (-2, 2)$
Insight

- We are interested in the enclosed area for \( \lambda > 0.5 \).
- When more items are comprised in \( S \), we search for better ranked objects in the area that corresponds to the intersection of all areas with improved objects \( \forall s_i \in S \).

Figure: Dividing curves for \( \lambda = 0.7 \)

(a) \( \vec{s} = (1, 0) \), \( \vec{p} = (1, 2) \)

(b) \( \vec{s} = (1, 1) \), \( \vec{p} = (1, 2) \)

(c) \( \vec{s} = (1, 1) \), \( \vec{p} = (-2, 2) \)