Distributed Similarity Search in High Dimensions

Sebastian Michel

Ecole Polytechnique Fédérale de Lausanne (EPFL)

sebastian.michel@epfl.ch

2009/03/10

Outline

Motivation

Similarity Search

- Query by example:
- Given an object, i.e., its "features".
- Find similar objects

Application Scenarios

- Search in audio/visual data
- Often characterized by a lot of dimensions

Query:

Result:

Motivation

Similarity Search

- Query by example:
- Given an object, i.e., its “features”.
- Find similar objects

Application Scenarios

- Search in audio/visual data
- Often characterized by a lot of dimensions

Query:

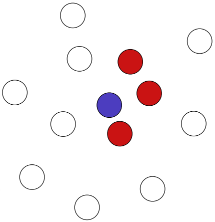
Result:

Similarity Search

Objects are characterized by a collection of relevant features, i.e., points in a high dimensional space.
We consider two kinds of queries:

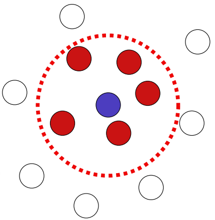
K-Nearest Neighbor (KNN) Queries

Given a query point q the goal is to find the K closest (in terms of the distance function) points to it.



Similarity Range Queries

Given a query point q and a range r the goal is to find all points within a distance r of q.



Our Work (Sketch)

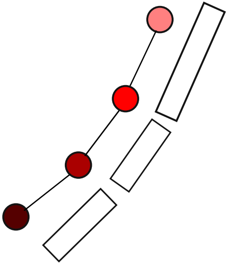
Distributed Similarity Search using Locality Sensitive Hashing (LSH)

Scenario

- Lots of machines maintain the index for similarity search
- Support of document granularity indexing and peer (user) granularity indexing

Key Idea

- Make use of the LSH method: hash similar objects to the same hash bucket
- Map the LSH bucket space to a linearly ordered set of nodes
- Mapping is locality preserving, too



Applications Scenarios

Document Indexing

- Users compute features of their objects and insert them into the distributed index

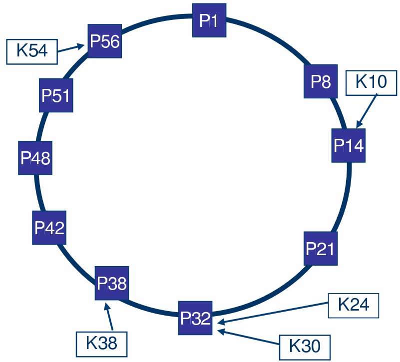
Peer/Collection Indexing

- Users compute representative objects (cluster centroids) on their local collection and index them
- Collections can be: personal documents, documents in a digital library, or per institute, etc
- At query time: KNN or range queries determine most promising collections to query in second step
- Actual query execution is performed locally at each selected peer

Distributed Hash Tables

Example: Chord

- Structured P2P overlay
- Map peers and keys to the same cyclic id space
- *lookup* method to find peer currently responsible for a key
- Naive Routing: Traverse successor links linearly
- Enhanced Routing: *Finger Tables* pointing to distant peers. Logarithmic cost



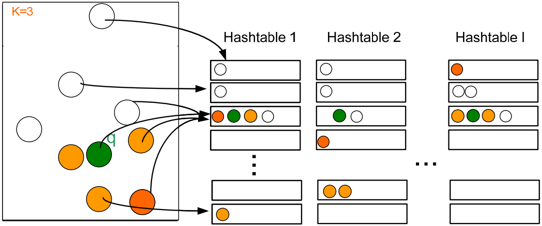
Locality Sensitive Hashing (LSH) [Indyk et al. STC 98]

Main Idea

- Collision based: map similar objects to same hash value
- Hash function selects bits from a binary object representation
- Hash value = bucket label
- Use several hash tables

	Table 1	Table 2
	0,1,0,1	**1,0,0,1**
	0,1,1,1	**1,0,0,1**
Result:	no match	match

What does the Locality Preserving Property mean?



Formally: A family of hash functions $H = \{h : S \rightarrow U\}$ is called (r_1, r_2, p_1, p_2)-sensitive if the following conditions are satisfied for any two points $\mathbf{q}, \mathbf{v} \in S$:

- if $dist(\mathbf{q}, \mathbf{v}) \leq r_1$ then $Pr_H(h(\mathbf{q}) = h(\mathbf{v})) \geq p_1$
- if $dist(\mathbf{q}, \mathbf{v}) > r_2$ then $Pr_H(h(\mathbf{q}) = h(\mathbf{v})) \leq p_2$

If $r_1 < r_2$ and $p_1 > p_2$: more similar objects are mapped to the same hash value than distant ones.

LSH variant [Datar et al. SCG 04]

for each d-dimensional data point $\mathbf{v}$ the hashing scheme considers k independent hash functions of the following form :

$$h_{\mathbf{a},B}(\mathbf{v}) = \left\lfloor \frac{\mathbf{a} \cdot \mathbf{v} + B}{W} \right\rfloor$$

where $\mathbf{a}$ is a d-dimensional vector whose elements are chosen independently from a p-stable distribution, $W \in \mathbb{R}$, and $B \in [0, W]$.

LSH bucket labels

With k hash functions, bucket label is integer vector of dimension k:

$$g(\mathbf{v}) = (h_{\mathbf{a_1},B_1}(\mathbf{v}), ..., h_{\mathbf{a_k},B_k}(\mathbf{v}))$$

	Table 1	Table 2
	15,1,55,9	25,34,11,94

Going Distributed (Observations&Requirements)

Large number of hash tables needed to achieve good accuracy.

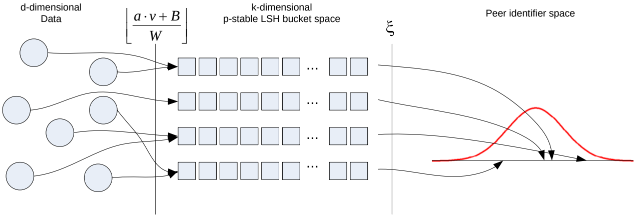
- Solution/approaches in centralized settings: probe multiple buckets, given rules to generate list of promising buckets [Panigrahy SODA 05, Lv et al. VLDB 07]

Distributed Settings

Multi-probe method seems to be infeasible, due to the large number of network hops. And, how many probes?

Linear Mappings

Recall: LSH hash value is a integer vector of dimension k.



We need a mapping from $\xi : \mathbb{Z}^k \to \mathbb{N}$ with the following desired properties:

Locality Preserving

Place similar objects on the same or neighboring peers.

Predictable Output Distribution

Needed for load balancing, peer placement.

LSH based on Linear Mappings

	Table 1	Σ
	15,1,55,9	80
	14,5,53,8	80
Result:	no match	match

Theorem

For any three points $\mathbf{v_1}, \mathbf{v_2}, \mathbf{q} \in S$ where $||\mathbf{q} - \mathbf{v_1}||_2 = c_1$ and $||\mathbf{q} - \mathbf{v_2}||_2 = c_2$ and $c_1 < c_2$ the following inequality holds:

$$pr(|h(\mathbf{q}) - h(\mathbf{v_1})| \leq \delta) \geq pr(|h(\mathbf{q}) - h(\mathbf{v_2})| \leq \delta)$$

Mappings

Output of LSH is a *k*-dimensional integer vector.

Considered Mappings ξ

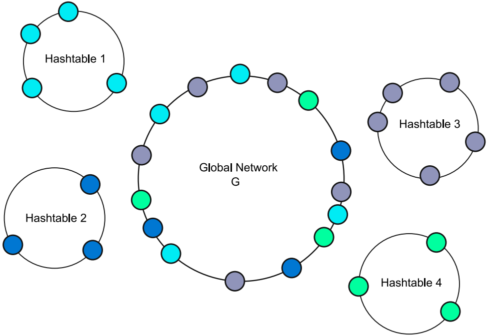
- summation (very intuitive): ξ_{sum}
- p-stable LSH (again): ξ_{LSH}

For both mappings we can estimate the distribution of the output.
$\rightarrow$ Needed for peer placement for effective load balancing.

Index Creation

Local and Global DHTs

- Global DHT G. All peers participate
- Local DHTs: one for each LSH hash table, M buckets.



Local Index Maintenance

First node (at position 1): responsible for $\mu_{sum} - 2 * \sigma_{sum}$ and the last bucket (at position M) to be responsible for $\mu_{sum} + 2 * \sigma_{sum}$:

$$\psi(value) := \left(\frac{value - (\mu_{sum} - 2 * \sigma_{sum})}{4 * \sigma_{sum}} * M\right) \bmod M$$

Index Creation (2)

Gateways to local DHTs

"Define" fixed position inside the local DHTs' value range. Drawn from samples, following known distribution. $\rightarrow$ more peers at the hot spots.

Gateways Selection

We select peers from the global DHT G based on

$$\rho(value, l) := (\psi(value) + hash(l))\ mod\ |G|$$

Handling Dynamics

Gateways Maintenance

Peer at $\rho(value, l)$ is not aware of being a gateway node. $\rightarrow$ it has to join the local DHT via another gateway node.

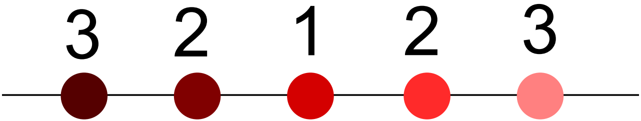
Growing Number of local Peers

If peers are overloaded: invite more peers from $|G|$ to local DHT.

Linear Forwarding

Initial Phase

1. For each hash table: select gateway peer
2. Forward query to gateway peer
3. Gateway peer forwards to responsible peer



Top-K Style Query Execution based on the locality sensitive mapping to the linear peer space by passing the query on to succeeding or preceding peers.

Stopping Condition

Stop if the best local result has a distance smaller than the current rank-k item.

Linear Forwarding (Algorithm)

- Natural stopping condition
- Fully exploits the placement function characteristics

```
LinearForward(query q, threshold τ, P_init, direction)
result[] = localIndex.executeLocalKnn(q)
if (result[0].distance> τ/α)
   done
else
   for(index=0; index< K; index++)
      if (results[index].distance< τ/α)
         resultSet.add(results[index])
      τ' = resultSet.rankKDistance()
   sendResults(resultSet, P_init)
   forwardQuery(...)
```

Range Query Processing

Initial Phase (Same as for KNN case)

1. For each hash table: select gateway peer
2. Forward query to gateway peer
3. Gateway peer forwards to responsible peer

Query is linearly forwarded, local range query execution at each peer.

Stopping Condition

Stop if there is no single item inside the specified range.

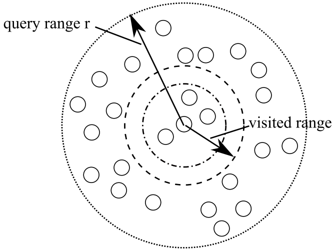
Processing Similarity Range Queries

Observation

Algorithm stops unnecessarily early due to holes in the considered range.

Solution

- Issue multiple concurrent requests at different peers
- Select peers based on sampling the potential query range



Method by Sahin et al. [DBISP2P '04]

Illustration of mapping a data point *v* to the peer space. The number of references is 8, the Chord range is $(0, 2^{10})$, each data point is replicated three times, w.r.t. different reference point positions.

$R = \{r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$

Sorted reference list (v) = $\{r_6, r_4, r_0, r_5, r_2, r_1, r_3, r_7\}$

Reference points to create Chord keys		Corresponding Chord keys
$(1^{st}, 2^{nd}) \longrightarrow$	$(r_6, r_4) \longrightarrow$	1101000000
$(1^{st}, 3^{rd}) \longrightarrow$	$(r_6, r_0) \longrightarrow$	1100000000
$(2^{nd}, 3^{rd}) \longrightarrow$	$(r_4, r_0) \longrightarrow$	1000000000

Experiments: Setup

Implementation

Implemented a simulation of the considered system.

Scenarios

- KNN queries, K=20
- range similarity search, varying ranges

Datasets

- **Flickr**: 1,000,000 images obtained from Flickr.com. MPEG7 visual descriptors. 282 dimensions, with descriptors such as *Edge Histogram Type* and *Homogeneous Texture Type*. 1,000,000 peers in the global DHT. Local DHTs of 1000 peers, each.
- **Corel**: 60,000 photo images from the Corel data set. 89 dimensions. Global DHT of 100,000 peers and 100 peers per local DHT.

Metrics and Opponents

- For both datasets, we use the Euclidean distance to measure the distances between points
- Queries: 100 randomly selected points from each dataset

Quality/Performance Measures

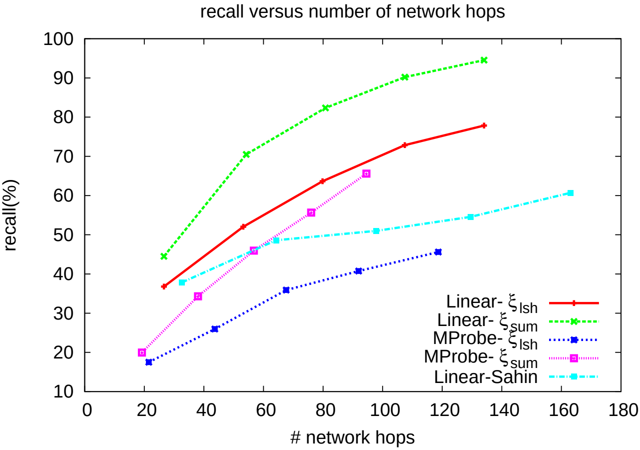
- Gini coefficient (i.e., indicator for the load distribution)
- Relative recall (i.e., precision w.r.t. true full-scan based answer)
- Error ratio: $\frac{1}{K}\sum_{i=1}^{K}\frac{d_{LSH_i}}{d_{true_i}}$
- Number of network hops

Opponents

- Simple LSH, no forwarding
- Method by Sahin et al.
- The LSH multi probe method
- Our KNN/range query methods

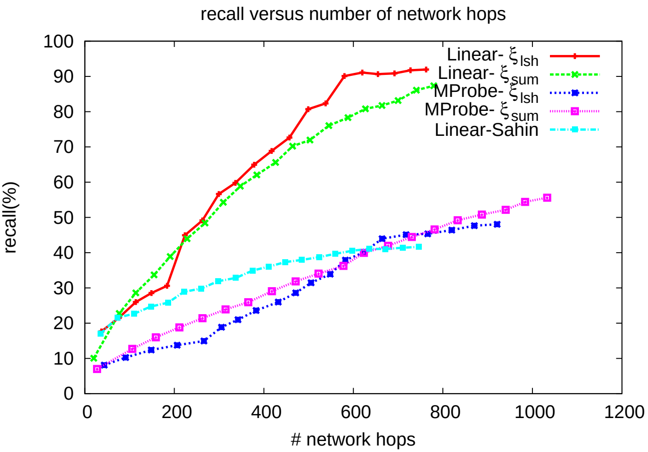
Recall vs. Number of Network Hops

For Corel



Recall vs. Number of Network Hops

For Flickr



Conclusion and Outlook

Conclusion

- Presented a novel approach for KNN and similarity queries in distributed environments
- Based on Locality Sensitive Hashing (LSH)
- Idea: Assign LSH hash buckets to a linearly ordered set of peers
- Placement is locality preserving, enables efficient KNN and range query processing
- Experiments show the applicability of the presented methods

Outlook

- The presented approach should be applicable to centralized settings, too, e.g., using an B+ tree index
- Approach of potential interest to cluster based systems

Danke
Thank you
Epharisto
Merci
Obrigado