An Adaptive Online System for Efficient Processing of Hierarchical Data

Athanasia Asiki
Dimitrios Tsoumakos
Nectarios Koziris
{nasia, dtsouma, nkoziris}@cslab.ece.ntua.gr
Motivation (1)

- **Efficient, on-line processing of bulk data**
  - Organized in concept hierarchies
  - Over one or more dimensions

- **Concept hierarchies**
  - A sequence of mappings from more general to lower-level concepts
  - Allow the structuring of information into categories

- **Observed in many applications**
  - Computer networks (e.g., router data)
  - Business (e.g., sales data)
  - Data warehouses
Example: Grid Information System

- Large-scale geographically distributed application by nature
- Large volumes of data
- Online update is required
  - Information generated continuously and at high rate
- Metadata of generated values follow concept hierarchies
- Peer-to-Peer technologies introduce:
  - Scalability
  - Fault-tolerance
  - Avoidance of single point of failures of centralized approaches
Grid Information System

VO

server

Hierarchical
structures

Registry

Information Producers

Information Consumers

An Adaptive Online System for Efficient Processing of Hierarchical Data

Athanasia Asiki
HPDC 2009
Grid Information System

VO server

Hierarchical structures

Tuples indexed with Germany by HPDC VO

Information Consumers

Tuples indexed by HPDC VO

Information Producers

Information Consumers

Registry

An Adaptive Online System for Efficient Processing of Hierarchical Data

Athanasia Asiki
HPDC 2009
Grid Information System

Queries concerning different levels of granularity:

- What is the Avg. CPU Time for a specific VO?
- Which is the Avg. CPU Time for a specific site?

But also:
- Which sites does a specific VO support?
Motivation (2)

- Exploitation of concept hierarchies
  - Organization of information on different levels of aggregated views
    - Efficient manipulation of data

- Provide a system to support hierarchical data

- Detection of real time changes in trends based on incoming queries

- Adaptive and flexible mechanisms based on the requirements of users
Our Goals

• Query data on different levels of granularity
  ▫ Roll-up towards more generalized levels
  ▫ Drill-down towards more detailed levels

• *Adaptive re-indexing on a per-tree basis according to the incoming queries*

• Maintenance of hierarchy specific information during store operations

• Online updates, while resolution of queries continues

• Distributed catalogue of stored data

• Support all above operations in a fully distributed environment
Roadmap

- Data insertion, while maintaining hierarchy-specific information

- Data Lookup in the DHT
  - With DHT lookups for values of the pivot levels
  - With soft-state indices
  - With flooding

- Re-indexing operations
  - Decision procedure
  - Roll-up / Drill-down

- Online updates

- Simulation results
Notation

Concept Hierarchy:
- VO
- Category
- Region
- Site

Example:
- VO: biomed
- Category: Production
- Region: SEE
- Site: HG-01

Same root keys:
- biomed
- Prod.
- SEE
- HG-01
- Norm. CPU Time: 181,198

Different pivot keys:
- biomed
- Prod.
- SEE
- HG-01
- Norm. CPU Time: 210,406

- biomed
- Prod.
- SWE
- CESGA-01
- Norm. CPU Time: 128,91
Insertion

• Look up of *root key*
• Node responsible for the *root key* :
  ▫ Find pivot key
  ▫ Create an index
• Store of tuple in the node responsible for its pivot key
• Tree structures
• Statistics per tree
Query processing

- **Exact-match queries**
  - Queries targeting pivot level
    - simple DHT lookups
  - Queries targeting root keys
    - use of indices

- **Flooded queries**
  - All the nodes scan their local databases
  - Soft-state, bidirectional indices from the node responsible for the queried value towards to the “actual” nodes are created

- **Indexed queries**
Re-indexing operations

- Re-indexing operations
  - Roll-up
  - Drill-down (Group Drill-down)

- Adapt the level of the indexing to the queries
  - on a per tree basis

- Re-indexing operations are triggered
  - After a flooded query
  - After a predefined number of queries for indexed values in a node

- Re-indexing towards a queried level (hence “most popular”):
  - if this level is the most popular
  - threshold% criterion
Drill-down

- Query for values belonging to levels below the pivot level
- If any level below this pivot level is the “most popular”, then drill down to this level

Scenario:
- Re-indexing decision after query for HG-01
- Drill-down to “Site” Level is decided
- Re-insertion of tuples with the new pivot keys
- Erasure of existing indices
- The root key (“biomed”) is informed about the new pivot keys
Drill-down

- Query for values belonging to levels below the pivot level
- If any level below this pivot level is the “most popular”, then drill down to this level
Roll-up operation

- Query for a value above the pivot level
- More than one nodes provide statistics
- If the queried level is the “most popular”, then the involved trees roll-up to this level
- Group Drill-down

**Scenario:**
- Query for “biomed”
- A node is positive to roll-up to the “VO” level
- The node that queried, collects statistic from all involved nodes and decides if a re-indexing operation is needed
Roll-up operation

- Query for a value above the pivot level
- More than one nodes provide statistics
- If the queried level is the “most popular”, then the involved trees roll-up to this level
- Group Drill-down

Before

After
Roll-up operation

- Query for a value above the pivot level
- More than one nodes provide statistics
- If the queried level is the “most popular”, then the involved trees roll-up to this level
- Group Drill-down
Update

- The appropriate pivot level should be selected

- If the pivot key exists
  - Selection of the used pivot level

- If the root key already but not the pivot key exists:
  - Selection of the existing MaxPivotLevel
  - Update of existing indices for values above the pivot level

<table>
<thead>
<tr>
<th>VO</th>
<th>Categ.</th>
<th>Reg.</th>
<th>Site</th>
<th>Norm. CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>esr</td>
<td>Tier2</td>
<td>GER.</td>
<td>DE-01</td>
<td>141,36</td>
</tr>
</tbody>
</table>

Biomed: SEE, SWE
esr: GER.
**Update**

- The appropriate pivot level should be selected

- If the pivot key exists
  - Selection of the used pivot level

- If the root key already but not the pivot key exists:
  - Selection of the existing `MaxPivotLevel`
  - Update of existing indices for values above the pivot level

<table>
<thead>
<tr>
<th>VO</th>
<th>Categ.</th>
<th>Reg.</th>
<th>Site</th>
<th>Norm. CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>esr</td>
<td>Tier2</td>
<td>GER.</td>
<td>DE-01</td>
<td>141,36</td>
</tr>
<tr>
<td>biomed</td>
<td>Prod</td>
<td>SEE</td>
<td>HG-03</td>
<td>95,6</td>
</tr>
</tbody>
</table>
Update

• The appropriate pivot level should be selected

• If the pivot key exists
  ▫ Selection of the used pivot level

• If the root key already but not the pivot key exists:
  ▫ Selection of the existing MaxPivotLevel
  ▫ Update of existing indices for values above the pivot level

<table>
<thead>
<tr>
<th>VO</th>
<th>Categ.</th>
<th>Reg.</th>
<th>Site</th>
<th>Norm. CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>esr</td>
<td>Tier2</td>
<td>GER.</td>
<td>DE-01</td>
<td>141.36</td>
</tr>
<tr>
<td>biomed</td>
<td>Prod</td>
<td>SEE</td>
<td>HG-03</td>
<td>95.6</td>
</tr>
<tr>
<td>biomed</td>
<td>Prod</td>
<td>FR</td>
<td>FR-01</td>
<td>302.3</td>
</tr>
</tbody>
</table>

![Diagram of an online system for efficient processing of hierarchical data.](image-url)
Experimental setup

- modified version of FreePastry DHT
- 256 nodes

**Synthetically generated data:**
- 100k tuples – 4-level concept hierarchy
- Value distribution
  - $|l_0|=100$
  - $|l_1|=1000 \Rightarrow Pivot\ level\ during\ initials\ insertions$
  - $|l_2|=10000$
  - $|l_3|=100000$
- Uniform distribution of values per level
- Each distinct value of level $l_i$ has a constant number of children in $l_{i+1}$
Experimental setup (2)

- **Queries**
  - 50k queries
  - Uniform or Zipfian distributions for the most popular levels
  - Uniform, 80/20, 90/10, 99/1 inside the level

- **Threshold% = 30%**

- **Performance metrics:**
  - **Precision**: percentage of queries answered without flooding
Skewed query workloads

- Better performance for more skewed workloads
- Limited number of distinct values in the upper levels
- Quicker to roll-up and adapt to the query workload

- Better performance for less skewed workloads inside the level
- Drill-down operations improve the precision
- Indices are not so useful due to the large number of distinct values
Multiple bias points

- Why is this challenging?
- Data is divided in quarters
- The most popular level is different for each quarter
- Over 92% of the queries are answered without flooding
- The number of roll-up and drill-down operations adjust to the query workload
Dynamic changes in skew

Dynamic change from $l_3$ to $l_o$

- Dynamic change in the skew of the workload
- Roll-up and drill-down operations contribute to the recovery of the achieved precision
- Existing indices are more useful, when the skew changes from $l_3$ to $l_o$ and thus the precision remains high, after the change in skew
- Precision is low when the skew changes from $l_o$ to $l_3$ until drill-down operations take place

Dynamic change from $l_o$ to $l_3$

- Precision is low when the skew changes from $l_o$ to $l_3$ until drill-down operations take place
Conclusions

- Mechanisms to store, index and query hierarchical data
  - Adaptive
  - Online
  - Distributed environment

- Use case: Grid IS

- Experimental evaluation
  - High precision in various types of workloads
  - Adaptiveness to incoming queries

- Future work
  - Implementation with multiple dimensions!
  - Implementation in a real testbed

Q&A

Thank you!