CoopSC
A Cooperative Database Caching Architecture

Andrei Vancea

Department of Informatics IFI, Communication Systems Group CSG, University of Zürich
Background

- Databases
  - Client / server architecture
  - Client: asks a query (SQL)
  - Server: returns the result (tuples)

- Data shipping architectures
  - Most of the query processing is performed on the client side
  - Data is sent from the server to the client at processing time
  - Client-side caching
    - Response time
    - Server load
Background

- Client-side caching
  - Page caching
  - Tuple caching
  - **Semantic caching**

- Semantic caching
  - Clients store the results of old queries, together with their descriptions
  - Old query results are used when answering new queries
Background - Semantic Caching

Example

Q1: select * from persons where age > 10
Q2: select * from persons where age > 18
Q3: select * from persons where age > 7

R1: age > 10

Semantic cache

R1: age > 10
Background - Semantic Caching

- Cache entry
  - Query description
  - Result set
- Query rewriting
  - Probe
  - Remainder
Cooperative Semantic Caching

- Share the local semantic caches between clients in a cooperative matter
- When answering a query, check if there are other clients that have useful semantic cache entries
- Why?
  - Reduce the load of the database
  - Decrease response time
Cooperative Semantic Caching

R1: age > 10

Q1: select * from persons where age > 10

result

Q3: select * from persons where age <= 10

result

select * from R1

result
Query Rewriting

- Query rewriting
  - Probe
  - Remote probes
  - Remainder
CoopSC

- Cooperatorative Semantic Caching
- Back-end: PostgreSQL
- Query Types
  - Selection (range predicates)
  - Projection
    - `select id, name, age from persons where 20 < age and age < 30`
- Cache organization
  - Semantic regions – stored locally by clients
  - Distributed index – built on top of a P2P overlay
- LRU (Least Recently Used) replacement policy
CoopSC

- Semantic regions
  - n-Tuples
  - Query description
    - Fields
    - Predicate

- Distributed Index
  - Indexes the semantic regions of all clients
  - Used for query rewriting
  - Built on top of Chimera P2P overlay
CoopSC - Architecture

- CoopSC GUI
- JDBC Driver
- CoopSC API
- SQL Parser
- Server Executer
- Peer Executer
- Query Executer
- Query Rewriter
- Distributed Index
- Cache Manager
- Chimera P2P Overlay
- In-Memory Storage

Client

Database Server

© 2010 UZH, IFI
```sql
select * from wisconsin where 10000 < unique1 and unique1 < 30000 and 10000 < unique2 and unique2 < 30000
```

### Union

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote Probe</td>
<td>127.0.1.1:2000</td>
</tr>
<tr>
<td>Select Project</td>
<td>wisconsin, unique1, unique2, two, four, ten, twenty, onepercent</td>
</tr>
<tr>
<td>Predicate</td>
<td>(20001 &lt;= unique1) and (unique1 &lt;= 29999) and (20001 &lt;= unique2) and (unique2 &lt;= 29999)</td>
</tr>
<tr>
<td>Region</td>
<td>1</td>
</tr>
<tr>
<td>Remote Probe</td>
<td>127.0.1.1:2005</td>
</tr>
<tr>
<td>Select Project</td>
<td>wisconsin, unique1, unique2, tenpercent, twentypercent, fiftypercent, unique3, evenonepercent, oddonepercent, string1, stringu2, string4</td>
</tr>
<tr>
<td>Predicate</td>
<td>(20001 &lt;= unique1) and (unique1 &lt;= 29999) and (20001 &lt;= unique2) and (unique2 &lt;= 29999)</td>
</tr>
<tr>
<td>Region</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>wisconsin</td>
<td>unique1, unique2, two, four, ten, twenty, onepercent</td>
</tr>
<tr>
<td></td>
<td>unique1, unique2, tenpercent, twentypercent, fiftypercent, unique3, evenonepercent, oddonepercent, string1, stringu2, string4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>wisconsin</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>(10001 &lt;= unique1) and (unique1 &lt;= 19999) and (10001 &lt;= unique2) and (unique2 &lt;= 19999)</td>
</tr>
<tr>
<td>Region</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>wisconsin</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>(10001 &lt;= unique1) and (unique1 &lt;= 19999) and (20000 &lt;= unique2) and (unique2 &lt;= 29999) or (20000 &lt;= unique1) and (unique1 &lt;= 20000) and (10001 &lt;= unique2) and (unique2 &lt;= 29999) or (20001 &lt;= unique1) and (unique1 &lt;= 20001) and (10001 &lt;= unique2) and (unique2 &lt;= 29999)</td>
</tr>
</tbody>
</table>

© 2010 UZH, IFI
Conclusion

- P2P approach used for reducing the load of the database server
- CoopSC Architecture