Niagara CQ: A Scalable Continuous Query System for Internet Databases

CPSC 504: DATA MANAGEMENT 2009
YONG

Outline
- Motivation
- What is NIAGARA CQ?
- What is Incremental Group Optimization?
- What is Query Split?
- Minor details + Performance
- Conclusion

Motivation
Continuous queries (CQ): allow users to receive new results when available.
Internet: large amount of frequently updating data.

CQs are popular & essential

Challenges
How can we manage millions of CQs to scale to the Internet most efficiently?

What is NIAGARA CQ?
- The Continuous Query sub-system of NIAGARA, which is a distributed database system for querying distributed XML data.
- Supports scalable continuous query processing

Niagara CQ: Novelty and Approaches
- Groups CQs based on similar query structure.
  - Grouped CQs share computation and data
    - reduce I/O
    - reduce unnecessary query invocations

Niagara CQ’s Grouping Technique
1) Incremental Group Optimization Strategy
2) Query Split Strategy
3) Uniform grouping of both time/change based queries

Niagara CQ Command Language
- CREATE CQ_name
- XML-QL query
- DO action
- {START start_time} {EVERY time_interval} {EXPIRE expiration_time}
- Delete CQ_name
**Incremental Group Optimization Strategy**

How do you group these continuous queries most efficiently???

- Groups are created for existing queries according to their signatures
  - Signatures = similar structures among the queries
- Groups allows the ‘common parts’ of queries to be shared
  - Common parts share result data from the ‘Group Plan’
- New query is merged into those existing groups that match its signatures.

**Expression Signature**

- Represent the same syntax structure, but possibly different constant values, in different queries.
- Expression signatures allow queries with the same syntactic structure to be grouped together to share computation.

**Group**

- Groups are created for queries based on their expression signatures. Consists of 3 parts:
  - Group signature: The common expression signature of all queries in the group.
  - Group constant table: The group constant table contains the signature constants of all queries in the group.

---

**Group (cont.)**

- Group plan: the group plan is the query plan shared by all queries in the group. It is derived from the common part of all single query plans in the group.
Discussion

- Expression signatures as described here are a very simple transformation. Are they too simple? That is, do they group together enough of the kinds of queries that this system is meant to handle?

- Do you think they would work better or worse for SQL queries instead of XML?

Incremental Grouping Algorithm

- When a new query is submitted:
  - Group optimizer traverses query plan bottom up to match its expression signature with the signatures of existing groups.
  - If no match, a new group will be generated.

Query Split Strategy

- How do we implement the destination buffer for ‘split operator’?
  1) Pipeline (BAD)
  2) Intermediate file (GOOD)

Pipeline buffer

- 1) Timer-based CQ... which tuple to store and for how long?
- 2) results in a single execution plan for all queries in the group
  - the query structure is a directed graph thus the plan may be too complicated
  - The combined plan can be very large
  - A large portion of the query plan may not need to be executed at each query invocation
  - Bottleneck

Materialized Intermediate Files

- Advantages
  - Each query is scheduled independently.
  - The potential bottleneck problem of the pipelined approach is avoided.

- Disadvantages
  - Extra disk I/Os.
  - Split operator becomes a blocking operator.
Other details

- Timer-based continuous queries fires at specific times, but only if the corresponding input files have been modified.
- Incremental evaluation allows queries to be invoked only on the changed data = ‘delta file’

Conclusion

NIAGARA CQ:
Incremental Group Optimization with Query Split

- scalable
- works better than non-groupings
- requires minimal change in query engine

Some performance comparisons

Discussion

- The authors motivate Niagara with a simple stock quote monitoring application. Is Niagara the best way to support this particular application? What other kinds of applications would Niagara be appropriate for?