Exploiting Common Subexpressions for Cloud Query Processing

Yasin N. Silva, Per-Ake Larson, Jingren Zhou

1 Arizona State University, 2 Microsoft Research, 3 Microsoft Corporation

Motivation

The Problem
- Cloud scripts often contain common subexpressions (CSEs).
- Initial aggregations are further aggregated or joined in several parts of the script.
- A conventional optimizer will produce a plan that evaluates these expressions multiple times.

Our Contribution
- We present a framework to correctly optimize cloud scripts that contain CSEs.
- CSEs are executed once and their results used by multiple consumers.
- The selection of the best plan is performed in a cost-based fashion.

A SCOPE Script

```sql
SELECT A, B, C FROM T1
USING C = FROM T2
GROUP BY A, B, C
```

Solution Overview

1. Identification of common sub-expressions
2. Optimize query recording physical properties of shared nodes
3. Propagate the information of shared nodes and identify LCAs
4. Re-optimize query enforcing physical properties at the shared nodes

1. Identifying CSEs
- Subexpression fingerprints are employed to quickly identify CSEs.
- A fingerprint is a highly compressed representation of a subexpression.

2. Recording Physical Properties
- At every shared node, we maintain the history of the physical properties for which an optimization task is created.
- The history of properties is stored as a linked list at every shared node.

3. Propagating information about shared groups and identifying LCAs
- The information about shared groups is propagated bottom-up from the shared groups to the root.
- The process also identifies, for each shared subexpression S, the least common ancestor group (LCA) of the consumers of S.

4. Re-optimizing the query enforcing physical properties
- This step re-optimizes the query enforcing physical properties at the shared groups.
- When an LCA node G is found, the process re-optimizes the subexpression rooted in G propagating a set of physical properties to be enforced in the CSE.

Handling Large Scripts

Exploiting Independent Shared Groups
- If multiple shared groups with the same LCA are independent, they can be re-optimized independently.

Example - Required rounds:
- \([p_1, q_1], [p_2, q_1], \ldots, [p_{15}, q_1]\)
- At this point we know best \(p_{(15,15)}\)
- \([p_{(15,15)}, p_{(15,15)}, \ldots, p_{(15,16)}]\)
- 15 rounds (instead of 64)

Performing Promising Rounds Early
- Shared groups are ranked based on potential repartitioning savings.
- Property sets are ranked based on the number of times they generated a best local plan during Phase 1.

Experimental Results

Example – Generated Plans

Conventional optimization
- Exploit Common Subexpressions

Experimental Evaluation Results

- Conventional Optimization
- Exploit Common Subexpressions