Exploiting Common Subexpressions for Cloud Query Processing

Yasin N. Silva[1], Per-Ake Larson[2] and Jingren Zhou[3]
[1]Arizona State University, [2]Microsoft Research, [3]Microsoft Corporation

Motivation

The Problem

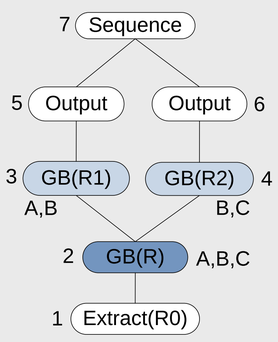
- Cloud scripts often contain common sub-expressions (CSEs).
- Initial aggregations are further aggregated or joined in several parts of the script.
- A conventional optimizers will produce a plan that evaluates these expressions multiple times.

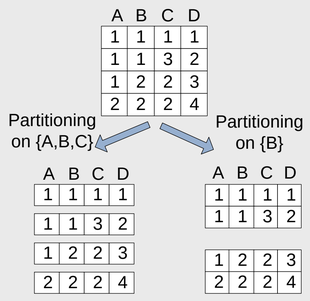
Our Contribution

- 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

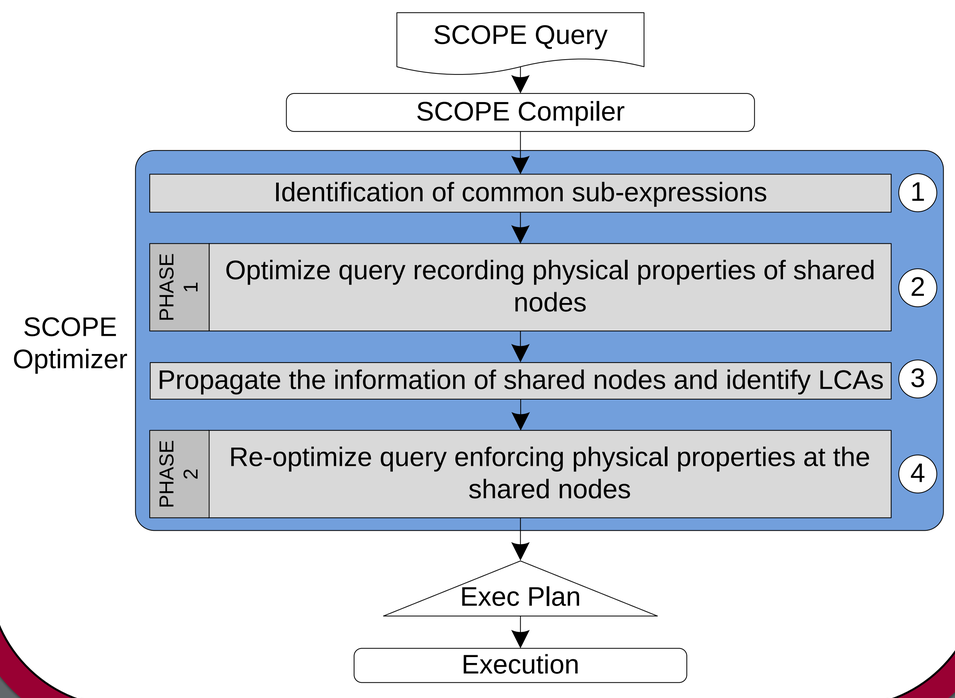
```
R0 = EXTRACT A,B,C,D,E FROM " ...\test.log "
USING LogExtractor;

R = SELECT A,B,C,Sum(D) as S FROM R0
    GROUP BY A,B,C;
R1 = SELECT A,B,Sum(S) as S1  FROM R
     GROUP BY A,B;
R2 = SELECT B,C,Sum(S) as S2 FROM R
     GROUP BY B,C;

OUTPUT R1 TO "result1.out";
OUTPUT R2 TO "result2.out";
```





Different ways to repartition the output of node 1

Solution Overview



1. Identifying CSEs

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

F_E, the fingerprint of expression E rooted in R is:

- If R is an operation that directly reads from a data file, then:

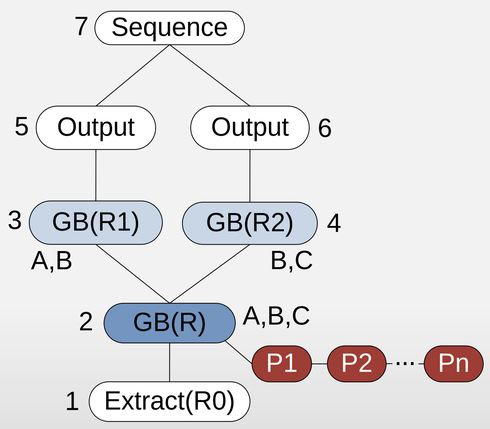
$$F_E = R.FileID \bmod N$$

- Otherwise,

$$F_E = (R.OpID \oplus (\bigoplus_{i=1}^{k} F_{R.child[i]})) \bmod N$$

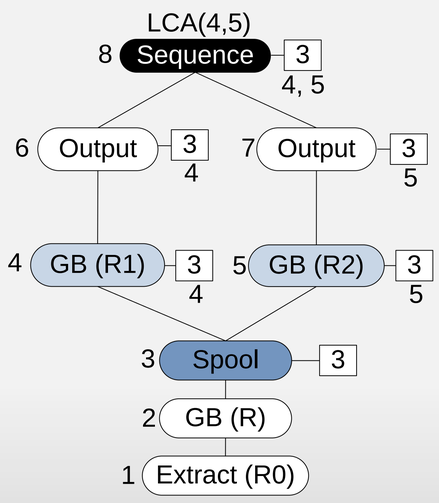
2. Recording Physical Properties

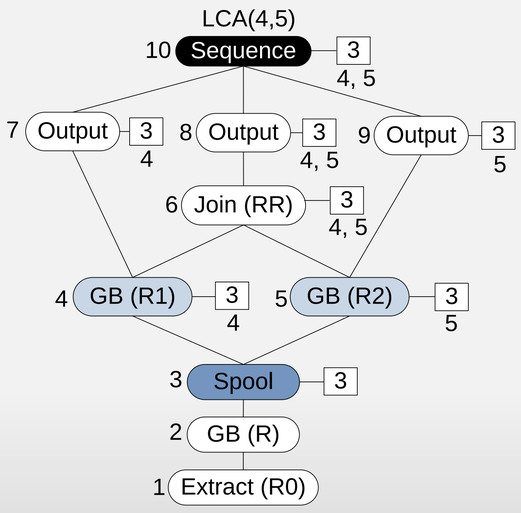
- At every shared node, we maintain the history of the physical properties for which an optimization task is created.
- The history of properties is stored as a linked list at every shared node.



3. Propagating information about shared groups and identifying LCAs

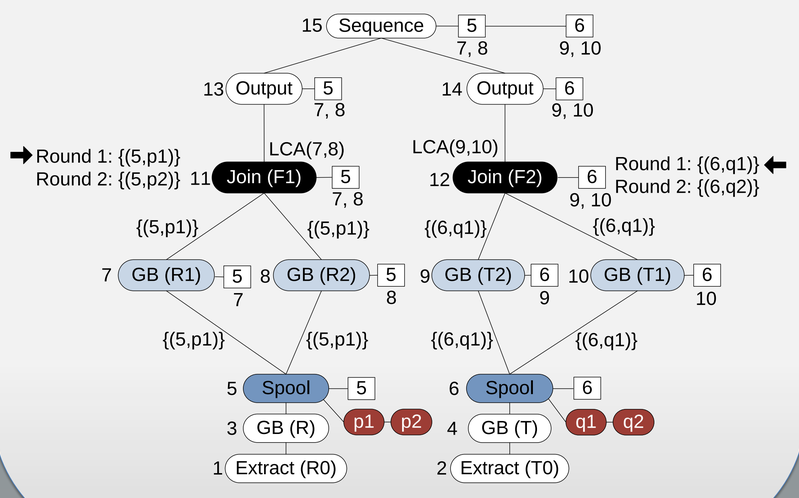
- The information about shared groups is propagated bottom-up from the shared groups to the root.
- The process also identifies, for each shared subexpression S, the least common ancestor group (LCA) of the consumers of S.





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.

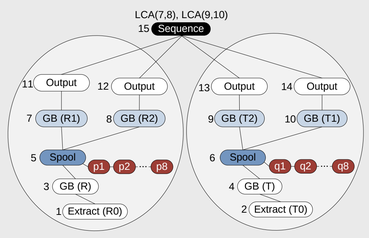
Example - Required rounds:

- {p1,q1}, {p2,q1}, …, {p8,q1}

At this point we know best p (p_{best})

- {p_{best},q2}, {p_{best},q3}, …, {p_{best},q8}

15 rounds (instead of 64)

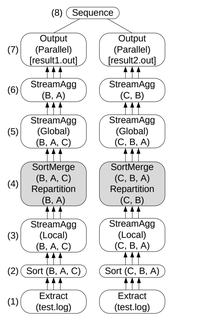


Performing Promising Rounds Early

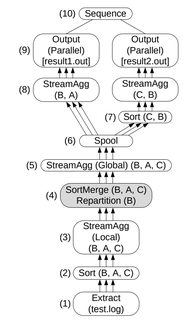
- Shared groups are ranked based on potential repartitioning savings.
- Property sets are ranked based on the number of times they generated a best local plan during Phase 1.

Experimental Results

Example – Generated Plans



Conventional optimization



Optimization exploiting common subexpressions

Experimental Evaluation Results