Peeking into the Optimization of Data Flow Programs with MapReduce-style UDFs

Motivation: Operator Reordering
- Data flow programming is a popular abstraction for complex analytics
- Diversity of data and tasks requires user-defined functions
- Operator order has significant impact on execution performance
- Reordering UDF operators requires knowledge of UDF properties

Context: Pact Programming Model

UDF Code Analysis
- Prerequisites:
  - Static Code Analysis Framework provides
  - Control-Flow, Def-Use, Use-Def lists
  - Fixed API to access records
- Extracted Information:
  - Field sets track read and write accesses on records
  - Upper and lower output cardinality bounds
- Safety:
  - All record access instructions are detected
  - Supersets of actual Read/Write sets are returned
  - Supersets allow fewer but always safe transformations

Data Flow Transformations
- Reorder Conditions:
  1. No Write-Read / Write-Write conflicts on record fields
  2. Preservation of groups for grouping operators
  3. Groups must remain unchanged or be completely removed

Enumeration Algorithm:
- Descents data flow recursively top-down
- Checks reorder conditions and switches successive operators

Defined Transformations:
- Filter push-down
- Join reordering
- Invariant group transformations
- Non-parallel operators are integrated

Physical Optimization
- Execution Plan Selection:
  - Chooses execution strategies for 2nd-order functions
  - Chooses shipping strategies to distribute data
  - Strategies known from parallel databases
- Interesting Properties:
  - Sorting, Grouping, Partitioning
  - Property preservation reasoning with write sets
- Cost-based Plan Selection:
  - Exploits UDF annotations for size estimates
  - Cost model combines network, disk I/O and CPU costs

Parallel Execution
- Execution Engine:
  - Massively parallel execution of DAG-structured data flows
  - Sequential processing tasks
  - Synchronous communication (in-memory and network)
- Runtime Operators:
  - Implemented as sequential processing tasks
  - Call UDFs

TPC-H Query 15 as PACT Program

CREATE VIEW revenue (supplier_no, total_revenue) AS
SELECT s_supplykey, SUM(l_extendedprice * (1 - l_discount)) AS total_revenue
FROM lineitem
WHERE l_shipdate < DATE '1998-01-01' + INTERVAL '3' MONTH
GROUP BY s_supplykey
PUBLIC void reduce(PactRecord records, Collector out) {
    PactRecord record = records.next();
    while (record != null) {
        record = record.next();
        out.collect(record);
    }
}