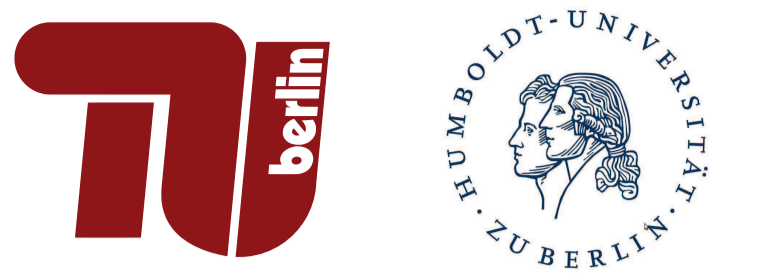


# Peeking into the Optimization of Data Flow Programs with MapReduce-style UDFs



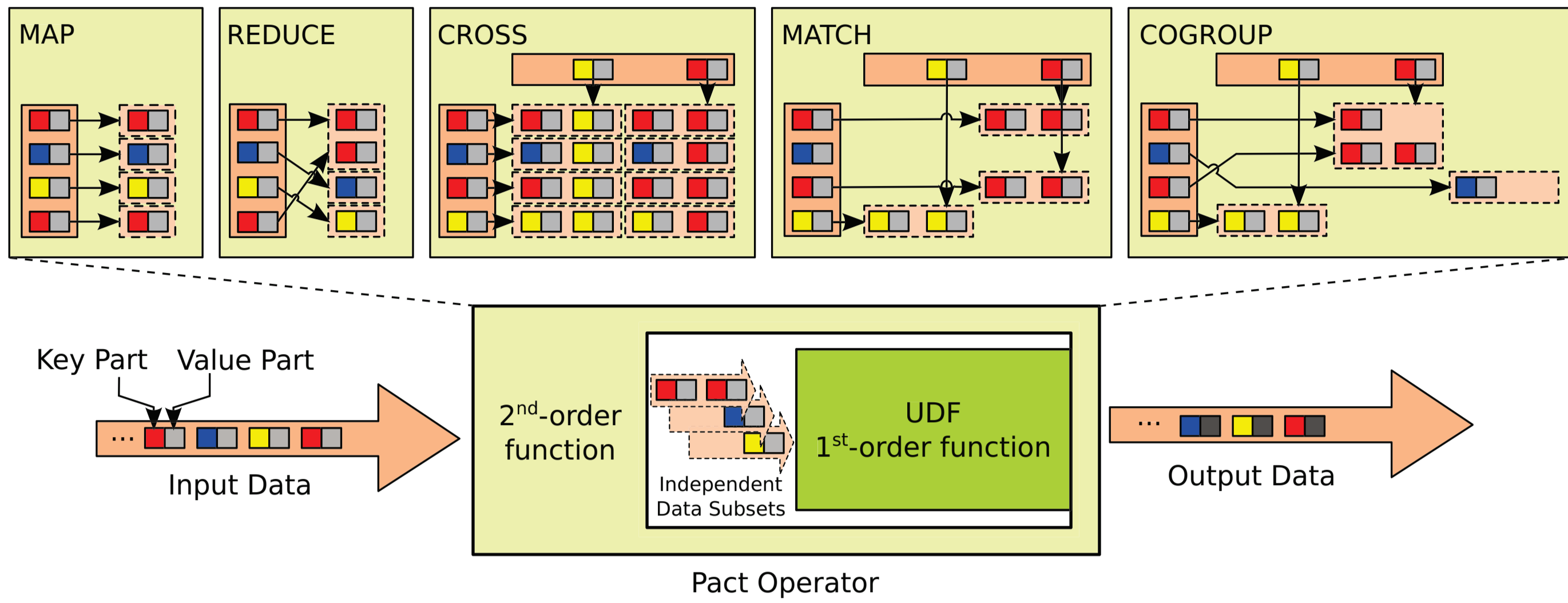
Fabian Hueske, Mathias Peters, Aljoscha Krettek, Matthias Ringwald, Kostas Tzoumas, Volker Markl, Johann-Christoph Freytag



## 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

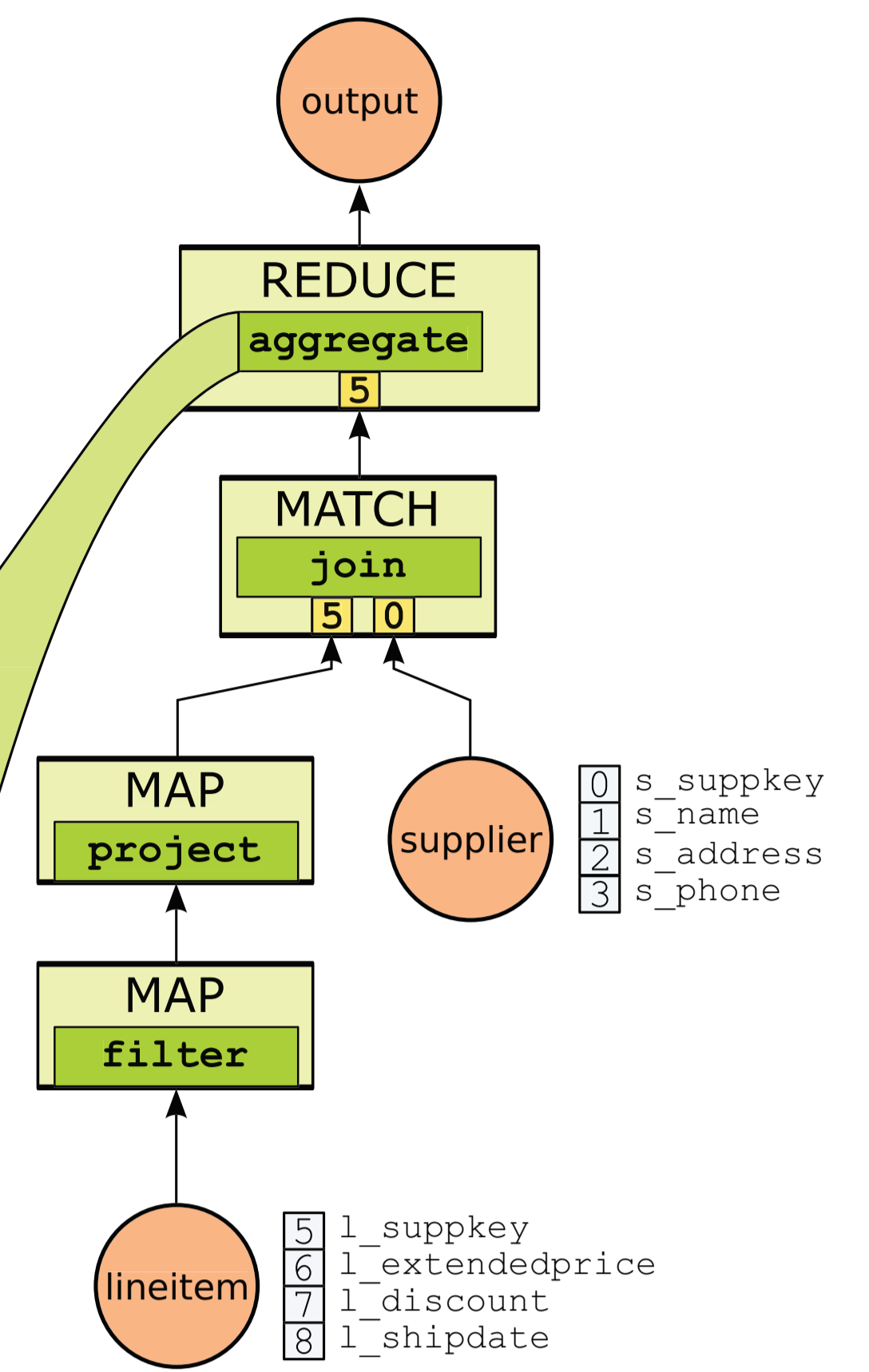


## TPC-H Query 15 as PACT Program

```
CREATE VIEW revenue (supplier_no, total_revenue) AS
SELECT l_suppkey, SUM(l_extendedprice * (1 - l_discount))
FROM lineitem
WHERE
  l_shipdate >= 'DATE' AND
  l_shipdate < DATE 'DATE' + INTERVAL '3' MONTH
GROUP BY l_suppkey;

SELECT s_suppkey, s_name, s_address, s_phone, total_revenue
FROM supplier, revenue
WHERE s_suppkey = supplier_no;
```

```
public void reduce(Iterator<PactRecord> records,
                  Collector out) {
  PactRecord resultRecord = records.next();
  double sum = resultRecord.getField(4);
  while (records.hasNext()) {
    resultRecord = records.next();
    sum += resultRecord.getField(4);
  }
  resultRecord.setField(4, sum);
  out.collect(resultRecord);
}
```



## UDF Code Analysis

### Prerequisites:

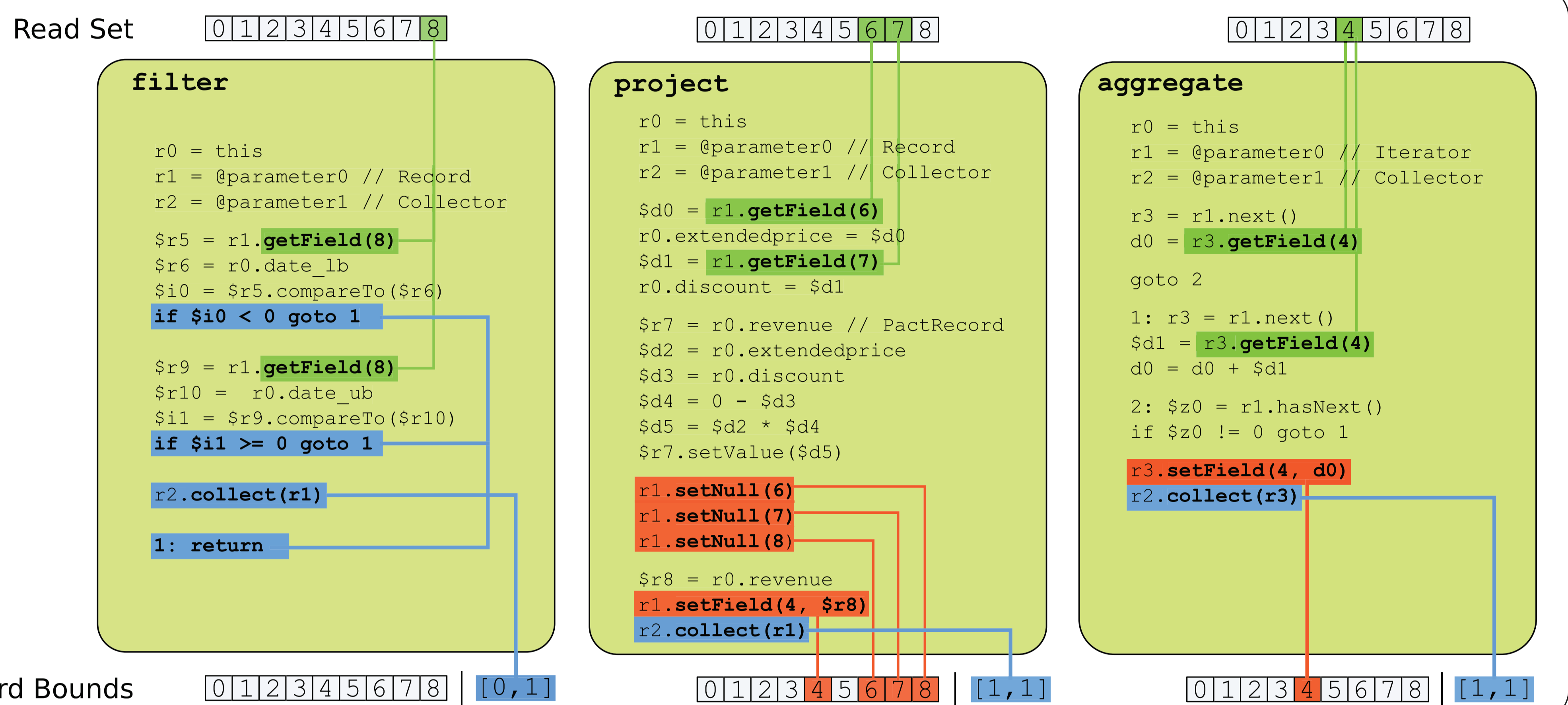
- Static Code Analysis Framework provides Control-Flow, Def-Use, Use-Def lists
- Fixed API to access records

### 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



Details in [HPS+12] and [HKT12]

Write Set | Out-Card Bounds

## Data Flow Transformations

### Reorder Conditions:

1. No Write-Read / Write-Write conflicts on record fields
  - Similar to conflict detection in optimistic concurrency control
2. Preservation of groups for grouping operators
  - Groups must remain unchanged or be completely removed

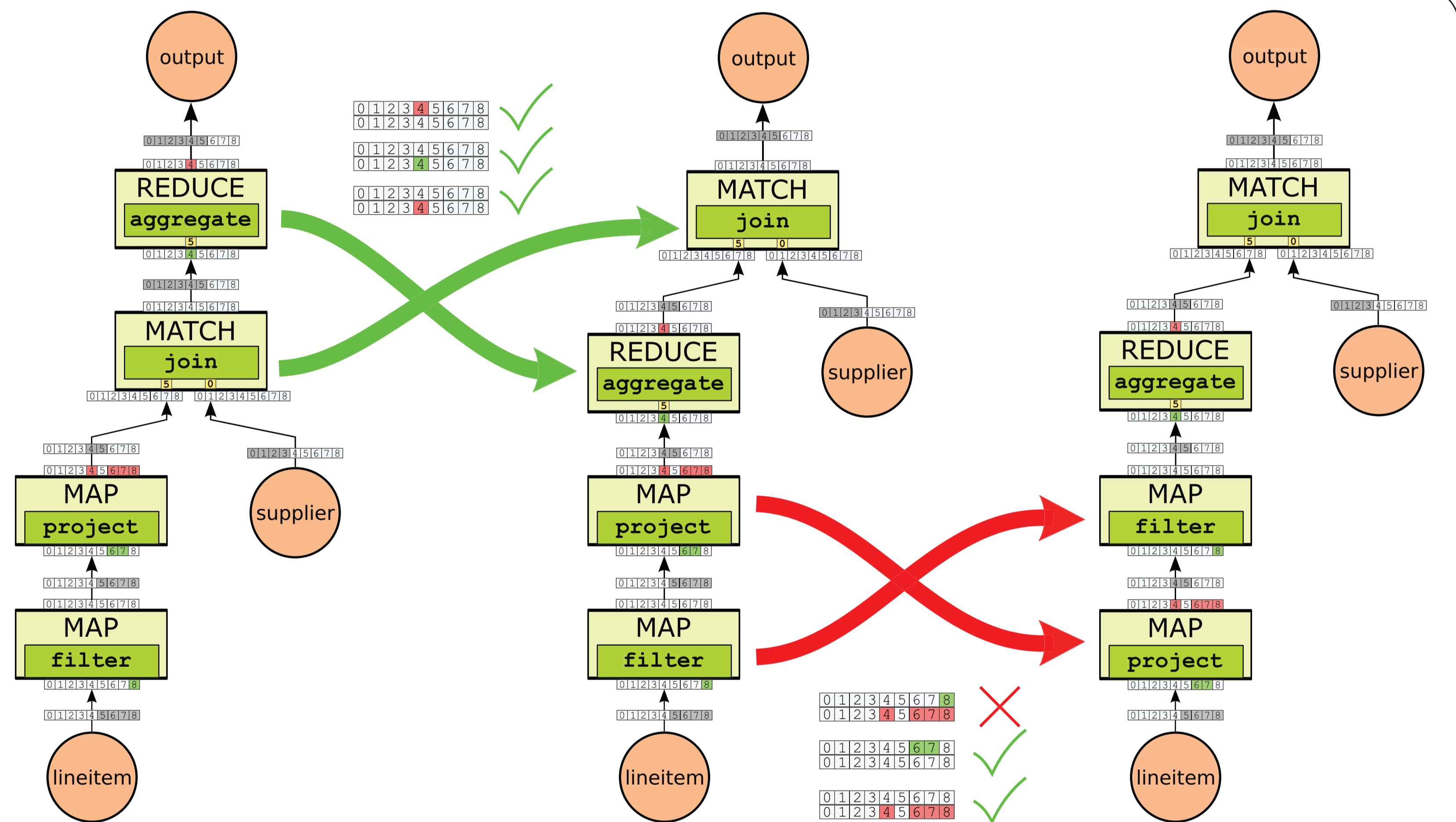
### Enumeration Algorithm:

- Descends data flow recursively top-down
- Checks reorder conditions and switches successive operators

### Supported Transformations:

- Filter push-down
- Join reordering
- Invariant group transformations
- Non-relational operators are integrated

Details in [HPS+12]



## Physical Optimization

### Execution Plan Selection:

- Chooses execution strategies for 2<sup>nd</sup>-order functions
- Chooses shipping strategies to distribute data
- Strategies known from parallel databases

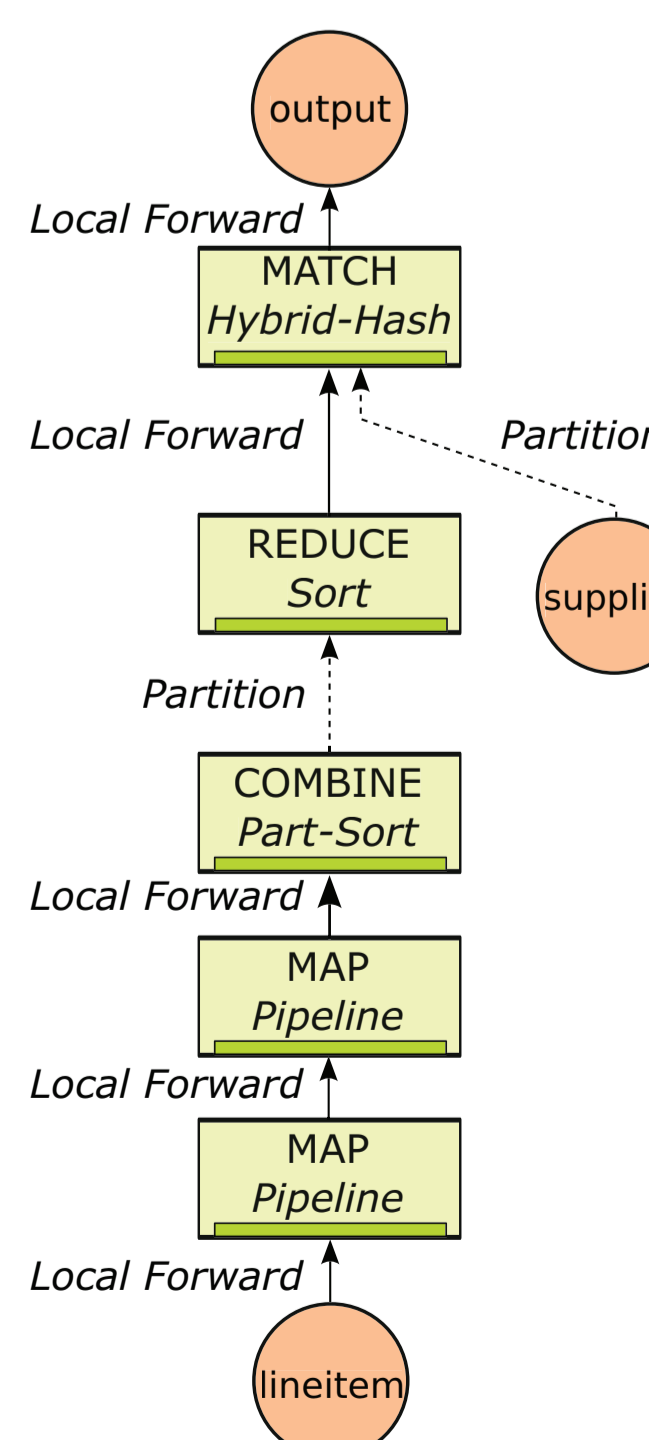
### Interesting Properties:

- 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

Details in [BEH+10]



## Parallel Execution

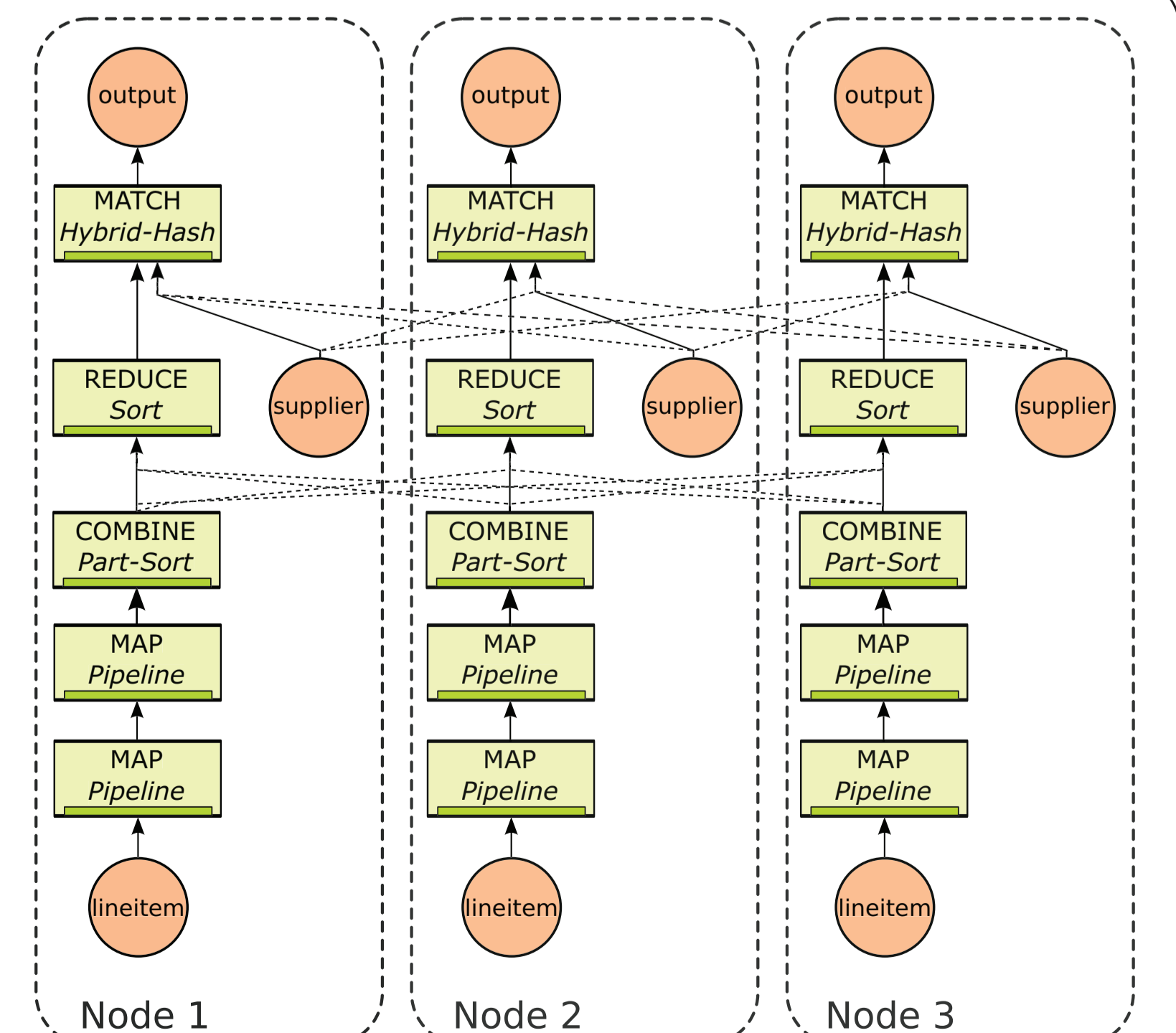
### Execution Engine:

- 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

Details in [BEH+10] and [WK09]



[WK09] Warneke, Kao, "Nephele: Efficient Parallel Data Processing in the Cloud", MTAGS '09

[BEH+10] Battré, Even, Hueske, Kao, Markl, Warneke, "Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing", SOCC '10

[HPS+12] Hueske, Peters, Sax, Rheinländer, Bergmann, Krettek, Tzoumas, "Opening the Black Boxes in Data Flow Optimization", PVLDB 5(11) '12

[HKT12] Hueske, Krettek, Tzoumas, "Enabling Operator Reordering in Data Flow Programs Through Static Code Analysis", XLDI Workshop '12

