EINSTEIN CENTER
Digital Future

The Stratosphere platform for big data analytics

Presentation by Daniyal Warsi, Timo Kraus & Sebastian Hofmann

Table of Contents

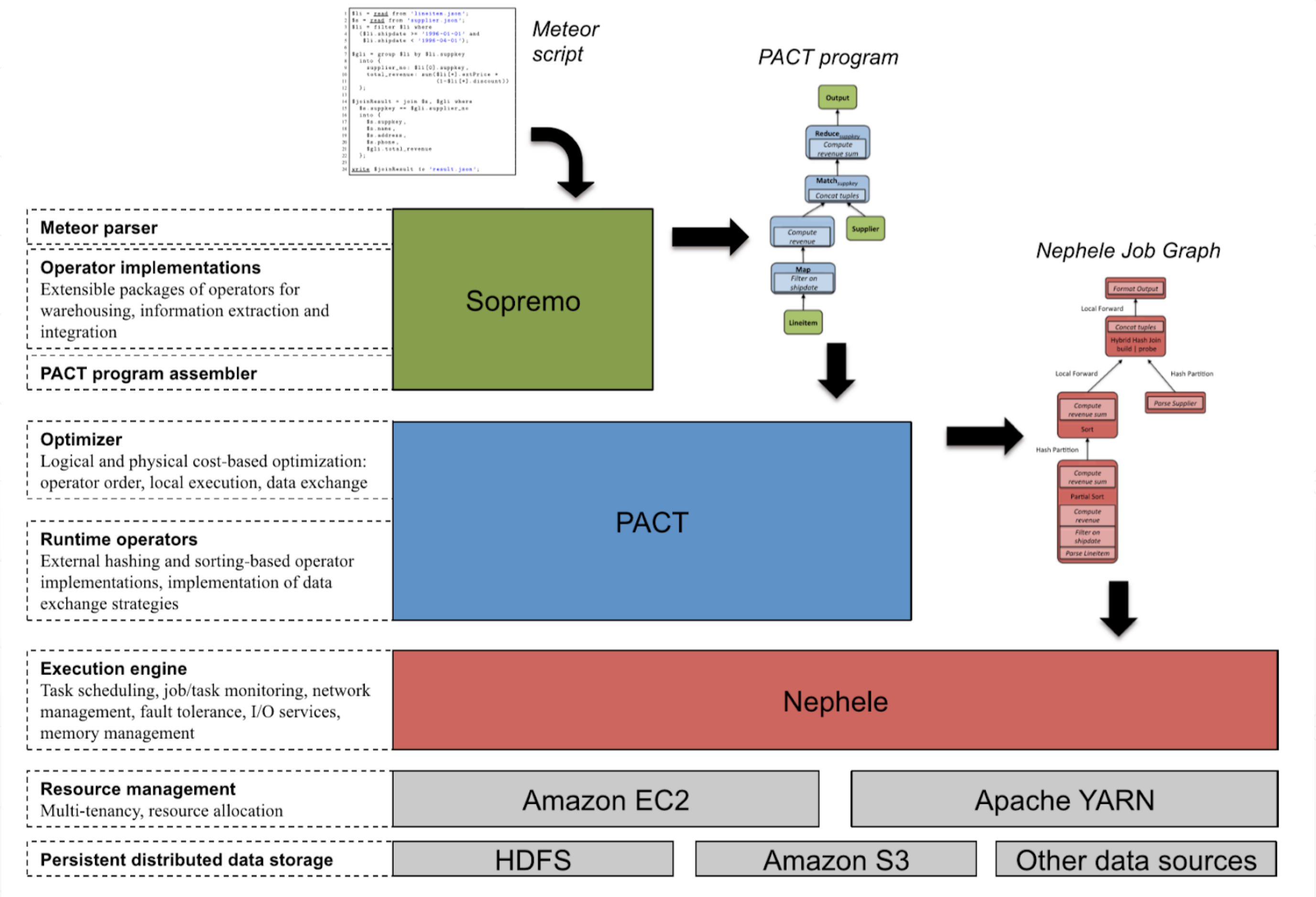
- Goal
- Stratosphere‘s Architecture
  - Sopremo
  - PACT
  - Nephele
- Fog Readiness
- IoT Use Case

Goal

The Stratosphere platform …

- provides a Big Data Analytics Platform that brings together high & low Level Programming.
- enhances traditional RDBMS with the Ability to cope with heterogenous Datasets.
- enables direct Connection to external Datasets.
- extends the MapReduce Operators for efficient Execution.

Stratosphere’s Architecture



```
$li = read from 'lineitem.json';
$s = read from 'supplier.json';
$li = filter $li where
  ($li.shipdate >= '1996-01-01' and
   $li.shipdate < '1996-04-01');

$gli = group $li by $li.suppkey
  into {
    supplier_no: $li[0].suppkey,
    total_revenue: sum($li[*].extPrice *
                       (1-$li[*].discount))
  };

$joinResult = join $s, $gli where
  $s.suppkey == $gli.supplier_no
  into {
    $s.suppkey,
    $s.name,
    $s.address,
    $s.phone,
    $gli.total_revenue
  };

write $joinResult to 'result.json';
```

Meteor parser

Operator implementations
Extensible packages of operators for warehousing, information extraction and integration

PACT program assembler

Optimizer
Logical and physical cost-based optimization: operator order, local execution, data exchange

Runtime operators
External hashing and sorting-based operator implementations, implementation of data exchange strategies

Execution engine
Task scheduling, job/task monitoring, network management, fault tolerance, I/O services, memory management

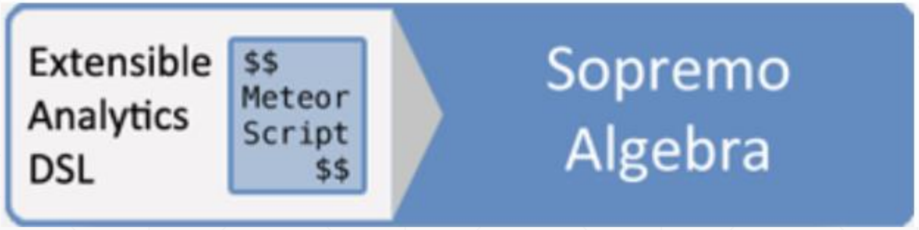
Resource management
Multi-tenancy, resource allocation

Persistent distributed data storage

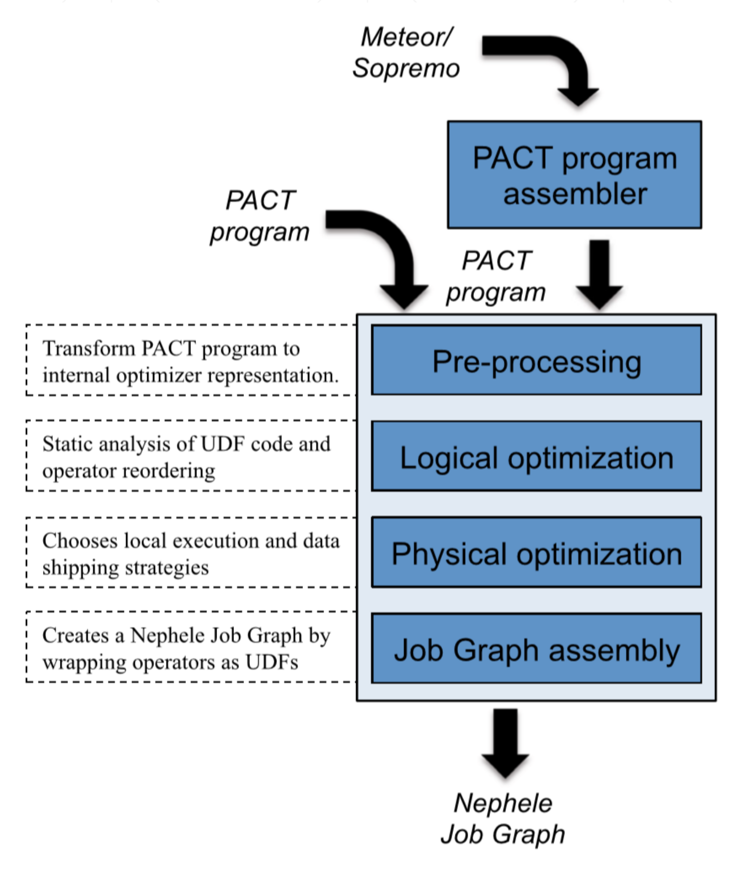
(Alexandrov et al., 2014, p.942)

Sopremo

- Declarative Query Language
- Meteor as the textual Interface for the Sopremo Layer
- Diverse set of composed Operators (Select, Project, etc.)
- Outputs a Logical Query Plan
- May be enriched with Metadata for lower levels



PACT

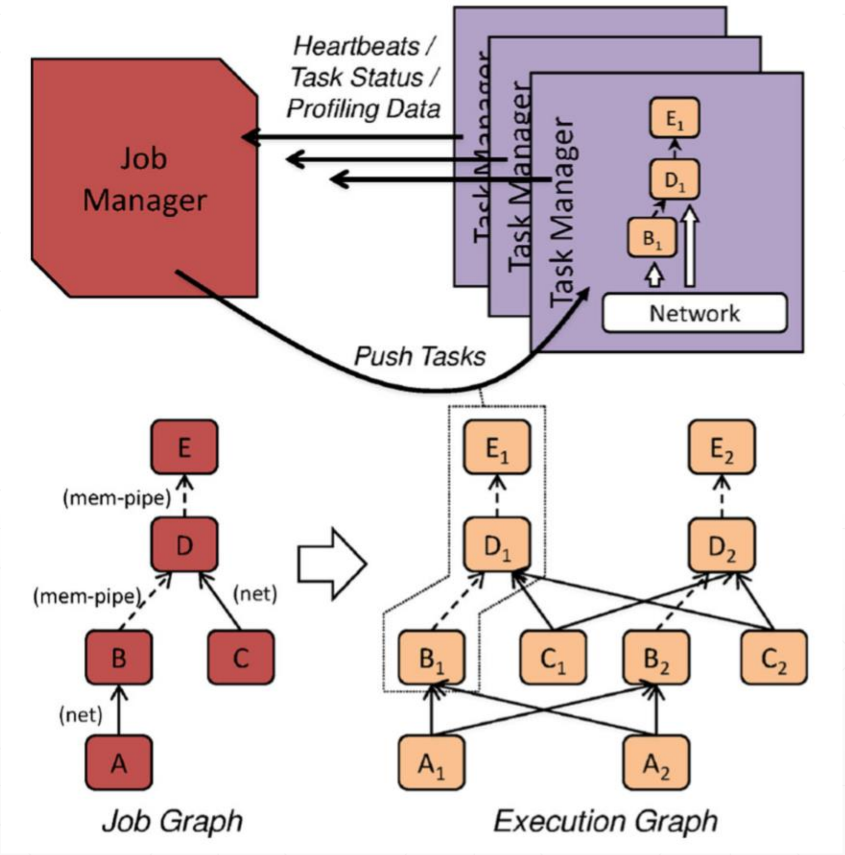

Meteor/
Sopremo
PACT program assembler
PACT
program
PACT
program
Transform PACT program to internal optimizer representation.
Pre-processing
Static analysis of UDF code and operator reordering
Logical optimization
Chooses local execution and data shipping strategies
Physical optimization
Creates a Nephele Job Graph by wrapping operators as UDFs
Job Graph assembly
Nephele
Job Graph

- Generalization of MapReduce
- Allows the Implementation of more complex Operators
- Includes a Query Optimizer for performant Execution
- Decomposes and reorganizes Operators
- Allows incremental Iterations for Pipelining

(Alexandrov et al., 2014, p.951)

Nephele

- Layer containing the Parallel Execution Engine
- Distributed Master/Worker Execution
- Includes Memory & I/O Services
- Collects Statistics for PACT Optimization



(Alexandrov et al., 2014, p.954)

Fog Readiness

Pros

- Nephele Master/Worker Distribution
- Bandwidth via Data Compression on Nodes
- Heterogenous Data Handling in IOT Use Cases
- Fault Tolerance via Buffering and Checkpointing

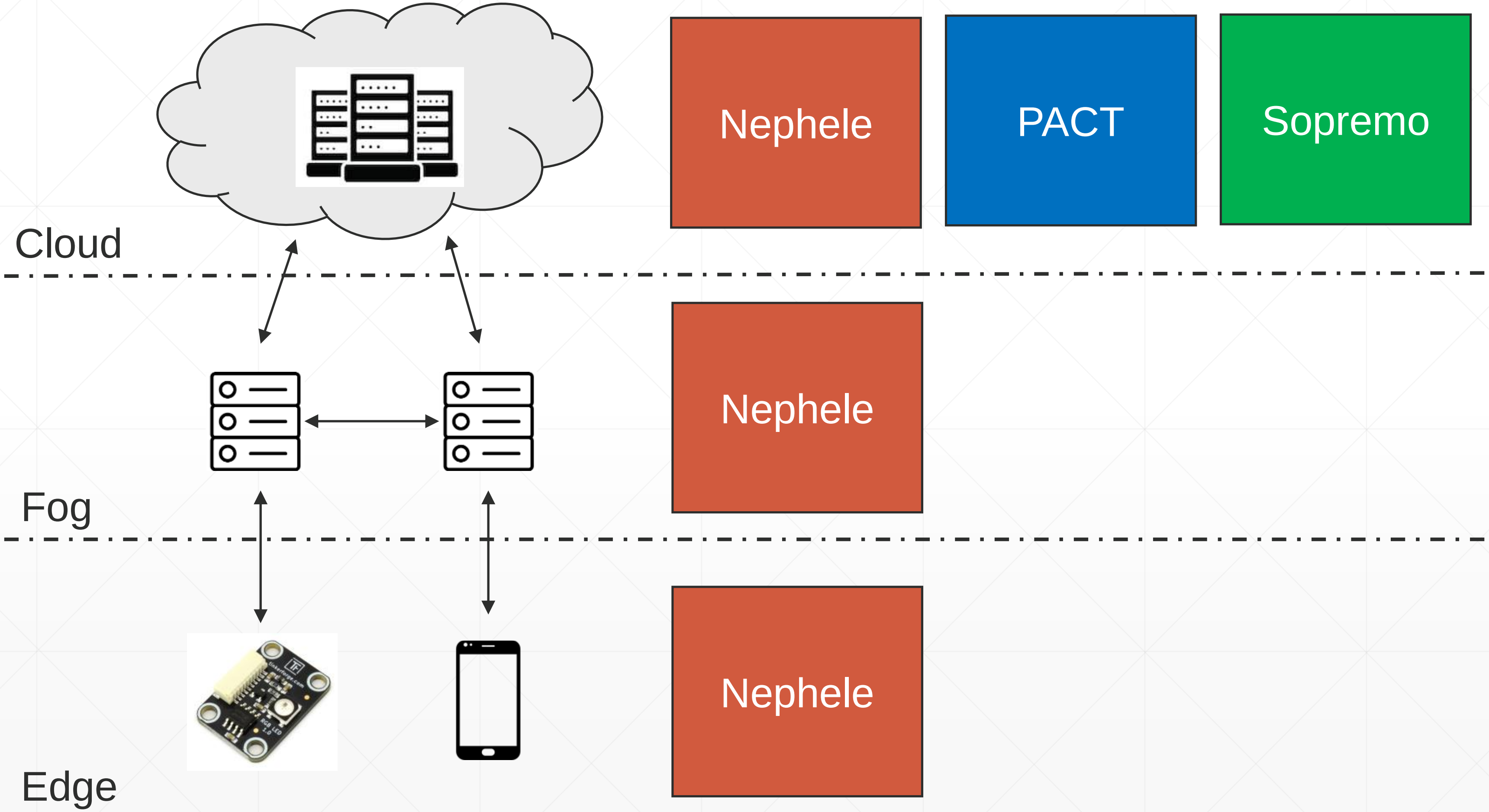
Cons

- Job Manager = Single Point of Failure
- Complexity in Query Optimization
- Testing on multiple Layers adds Complexity

Neutral

- Nephele Parallelization supports Custom Events → Implementation for Geo-Distribution
- Security Aspects not discussed

Stratosphere in an IoT Use Case



- Data Analytics Computation
- Pact Optimization
- Nephele Parallelization & Geo-Distribution

- Nephele Master Nodes
- Data Compression

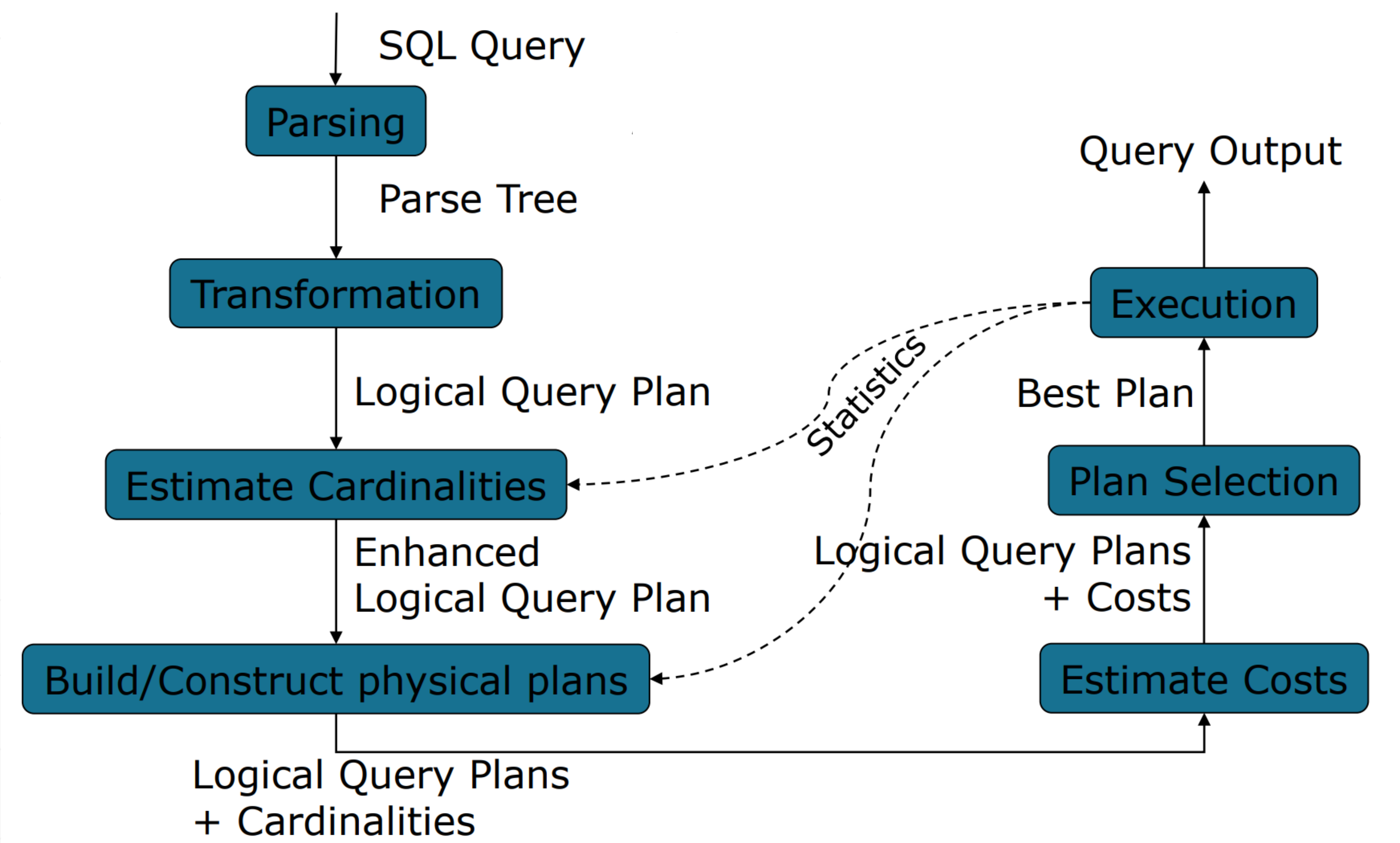
- Nephele Worker Nodes
- Data Generation
- Visualization of Results

Sources & Related Work

- **Original Paper:** Alexandrov et al. (2013): The Stratosphere platform for big data analytics
- Dean, Ghemawat (2004): MapReduce: Simplified Data Processing on Large Clusters
- Ang, Seng (2016): Big Sensor Data Applications in Urban Environments
- Shvachko et al. (2010): The Hadoop Distributed File System
- Rabl (2018): Internals of Database Systems: Web-Scale Data Management - Analytical Processing

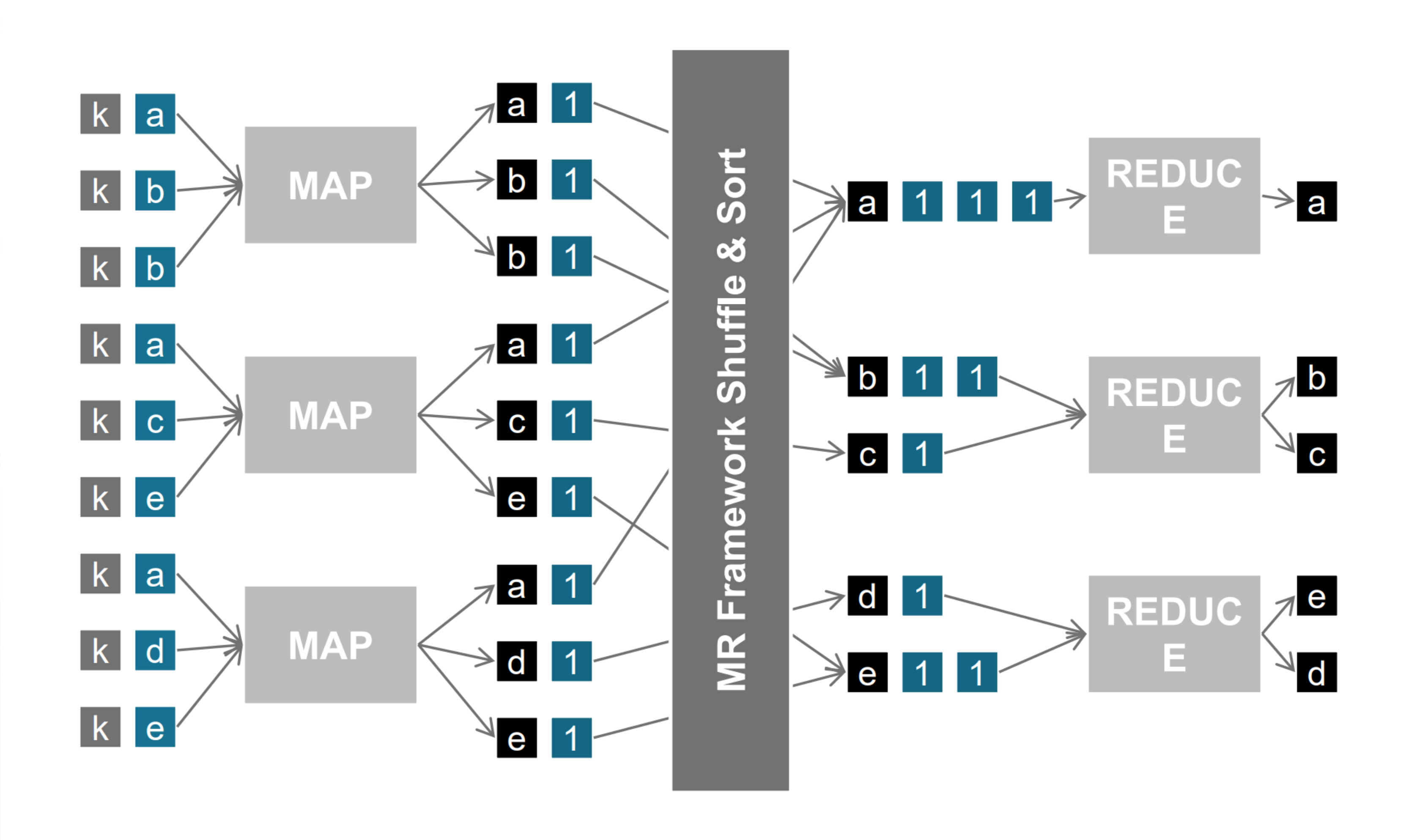
Annex

Query Processing



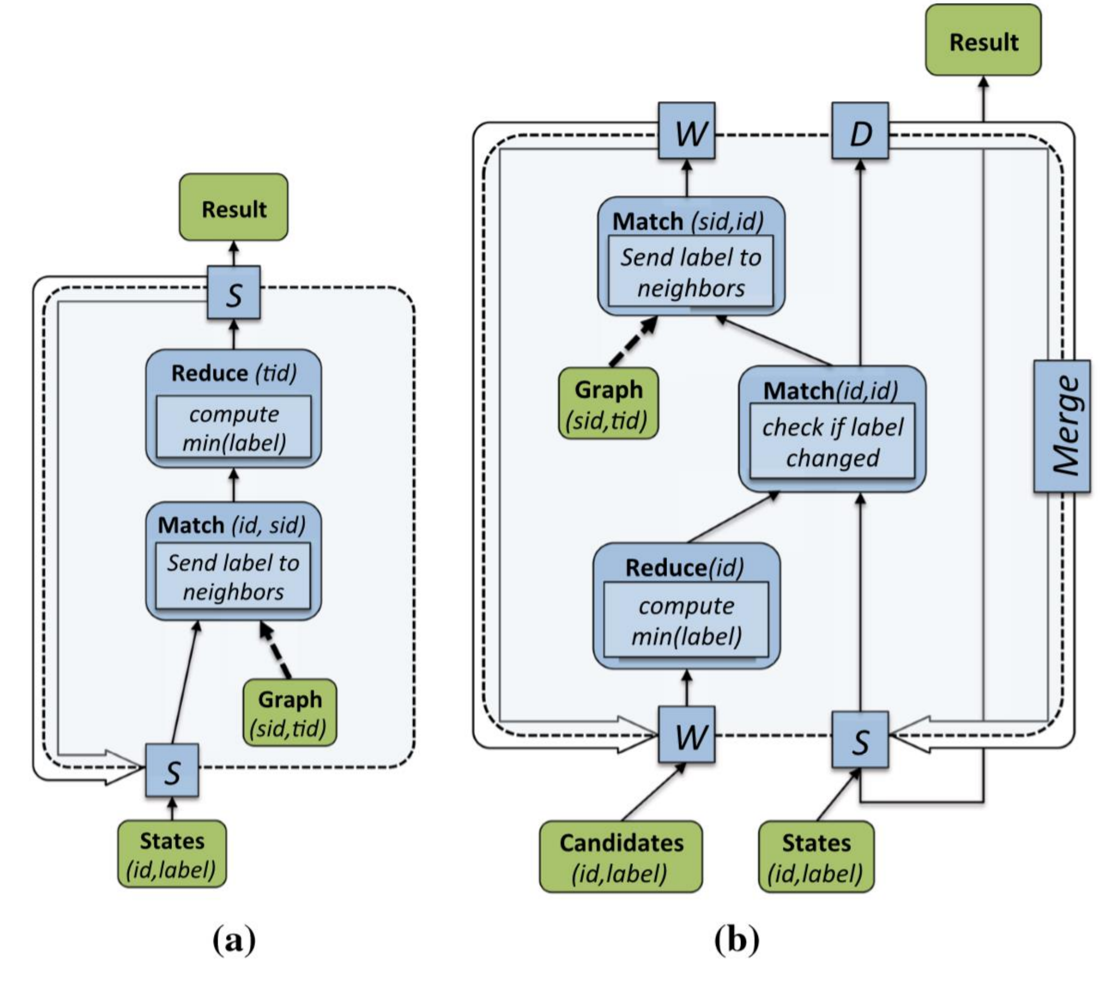
(Rabl, 2018, p.4)

MapReduce



(Rabl, 2018, p.18)

Bulk vs. incremental Iteration


Result
S
Reduce (tid)
compute min(label)
Match (id, sid)
Send label to neighbors
Graph (sid,tid)
S
States (id,label)
(a)
Result
W
D
Match (sid,id)
Send label to neighbors
Graph (sid,tid)
Match(id,id)
check if label changed
Merge
Reduce(id)
compute min(label)
W
S
Candidates (id,label)
States (id,label)
(b)