Zoi Kaoudi
Sebastian Kruse
Jorge Quiané

RHEEM @ BOSS

— Turning a Zoo into a Circus —

معهد قطر لبحوث الحوسبة
Qatar Computing Research Institute

جامعة حمد بن خليفة
HAMAD BIN KHALIFA UNIVERSITY

مؤسسة قطر
Qatar Foundation

HPI

Let’s Start with 2 Facts

Nature Inspired

Go fast?

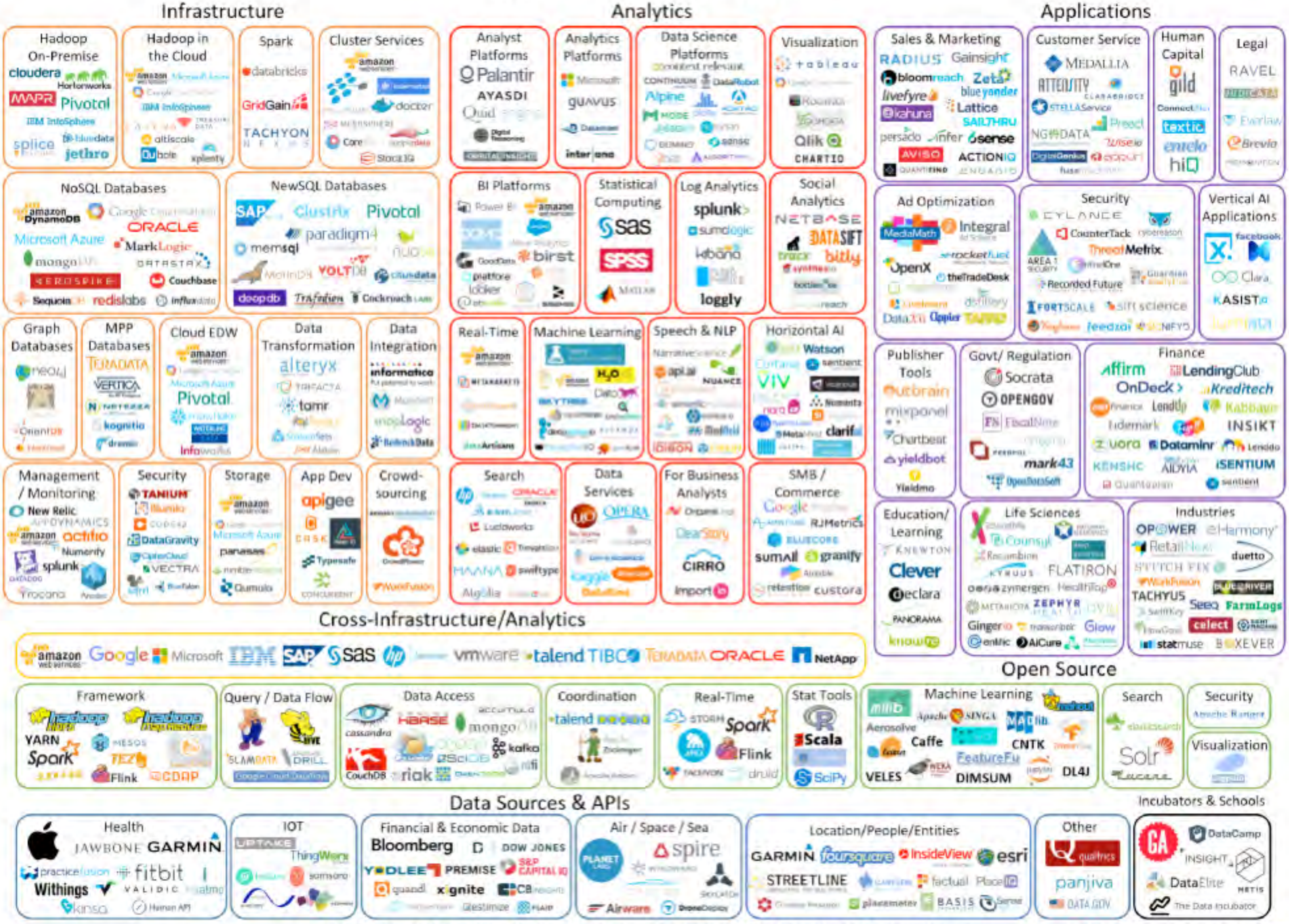
Endurance?

Go to the forest?

FACT 1:

One Size **Doesn’t** Fit All

Big Data Landscape 2016



Infrastructure

Analytics

Applications

Hadoop On-Premise

Hadoop in the Cloud

Spark

Cluster Services

Analyst Platforms

Analytics Platforms

Data Science Platforms

Visualization

Sales & Marketing

Customer Service

Human Capital

Legal

NoSQL Databases

NewSQL Databases

BI Platforms

Statistical Computing

Log Analytics

Social Analytics

Ad Optimization

Security

Vertical AI Applications

Graph Databases

MPP Databases

Cloud EDW

Data Transformation

Data Integration

Real-Time

Machine Learning

Speech & NLP

Horizontal AI

Publisher Tools

Govt/ Regulation

Finance

Management / Monitoring

Security

Storage

App Dev

Crowd-sourcing

Search

Data Services

For Business Analysts

SMB / Commerce

Education/ Learning

Life Sciences

Industries

Cross-Infrastructure/Analytics

Open Source

Framework

Query / Data Flow

Data Access

Coordination

Real-Time

Stat Tools

Machine Learning

Search

Security

Visualization

Data Sources & APIs

Incubators & Schools

Health

IOT

Financial & Economic Data

Air / Space / Sea

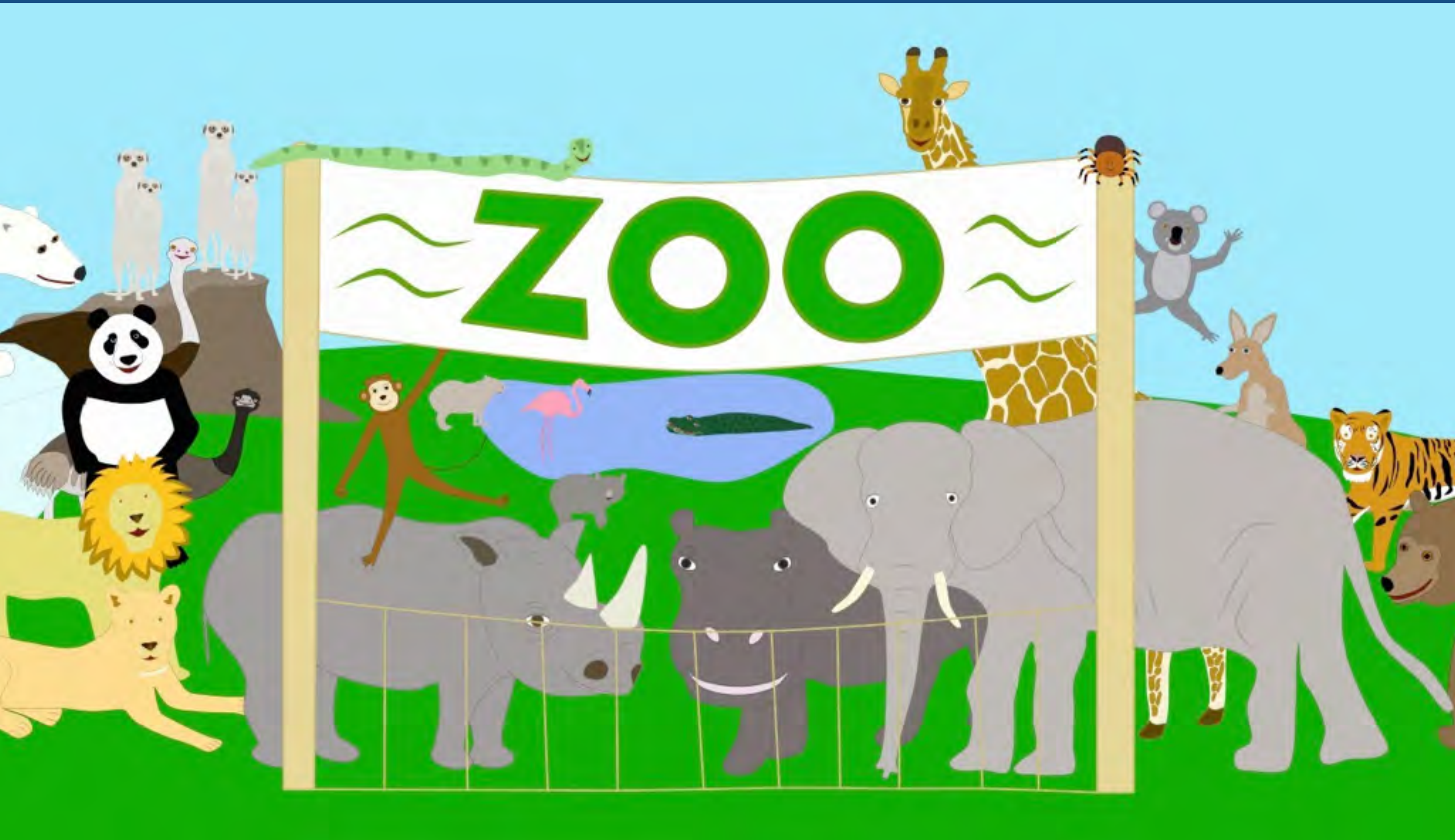
Location/People/Entities

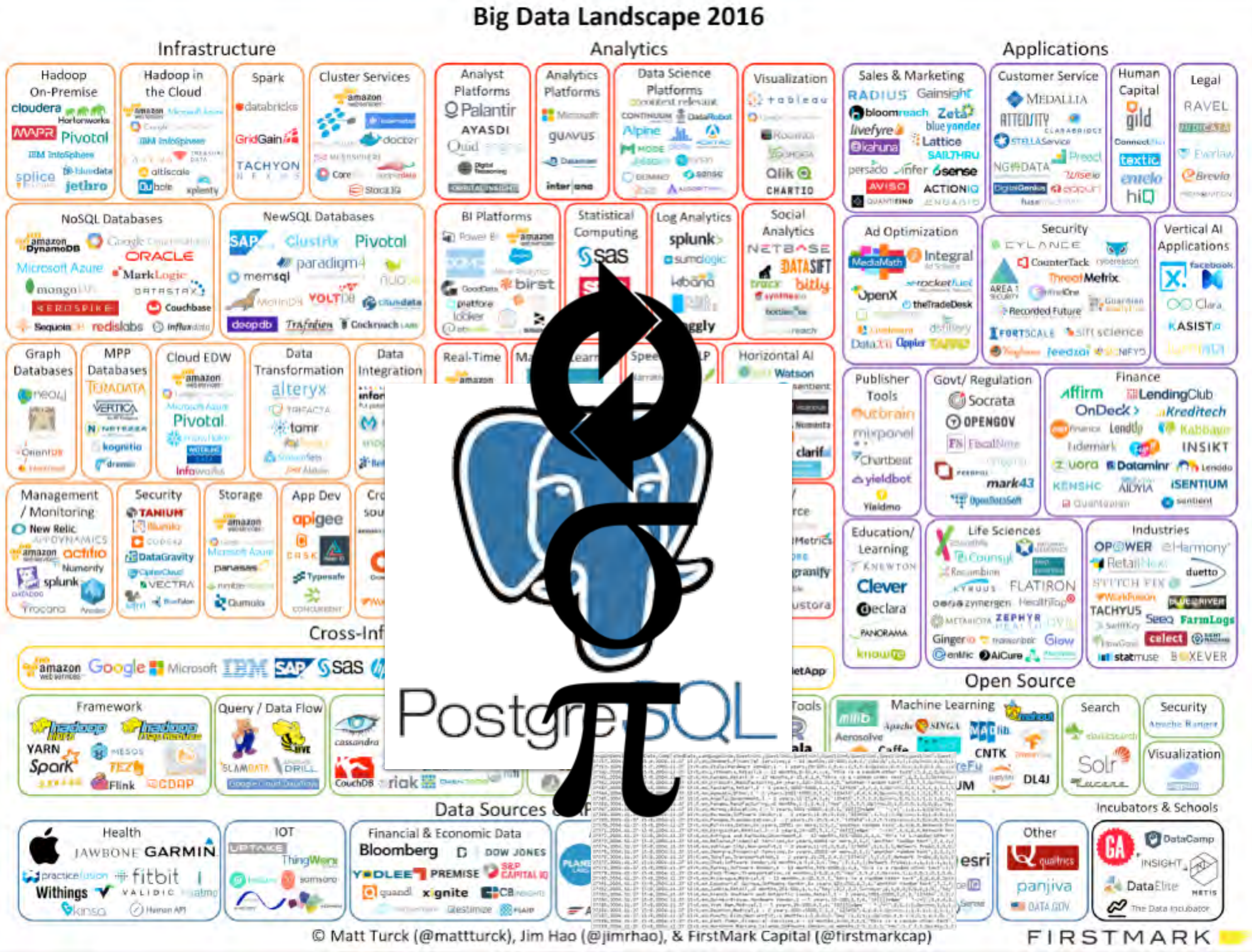
Other

© Matt Turck (@mattturck), Jim Hao (@jimrhao), & FirstMark Capital (@firstmarkcap)

FIRSTMARK

FACT 2: Zoo of Systems





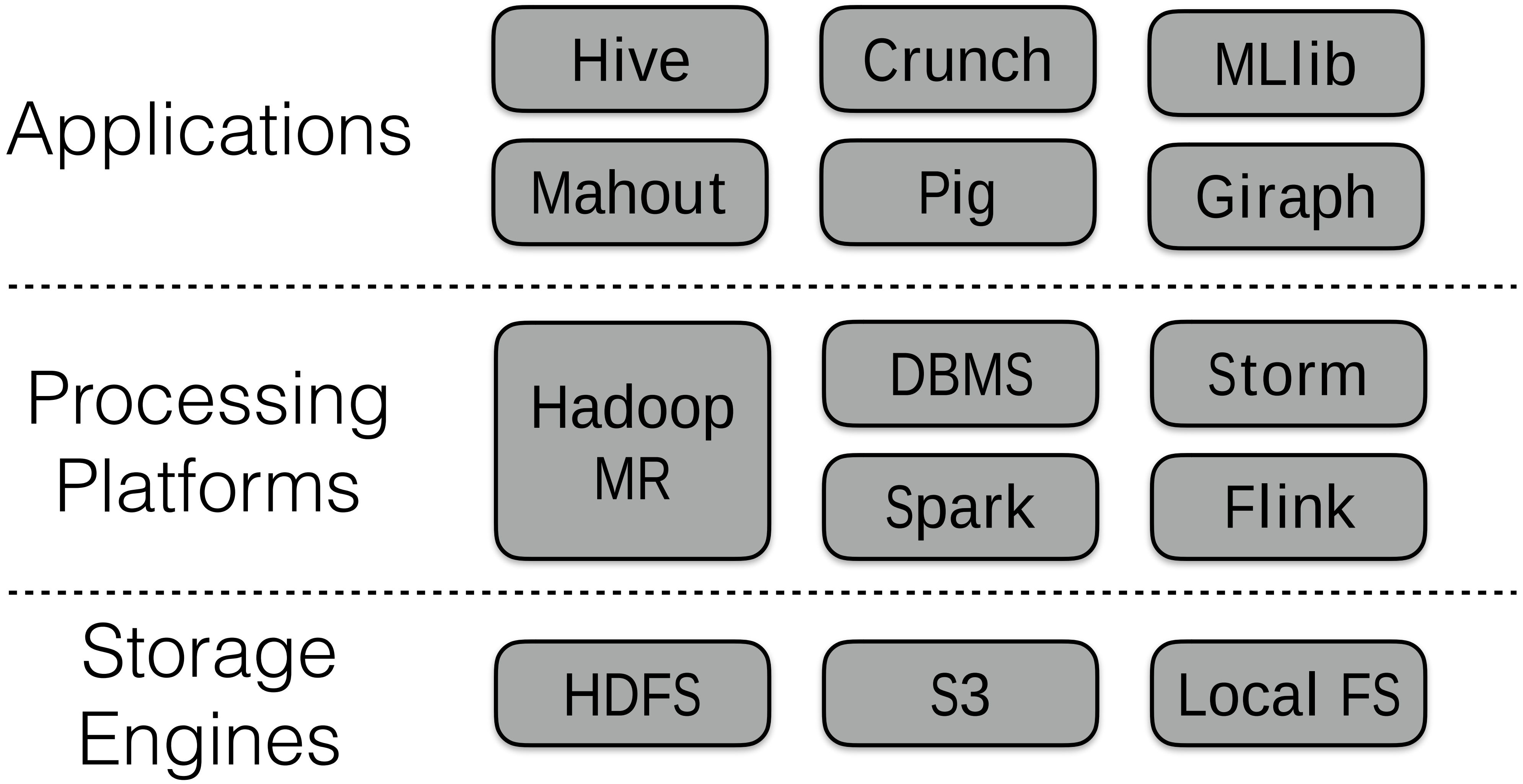
Big Data Landscape 2016

© Matt Turck (@mattturck), Jim Hao (@jimrhao), & FirstMark Capital (@firstmarkcap)

What is RHEEM?

A System Tamer

Where in the Analytics Stack?



Applications

Hive

Crunch

MLlib

Mahout

Pig

Giraph

Processing Platforms

Hadoop MR

DBMS

Storm

Spark

Flink

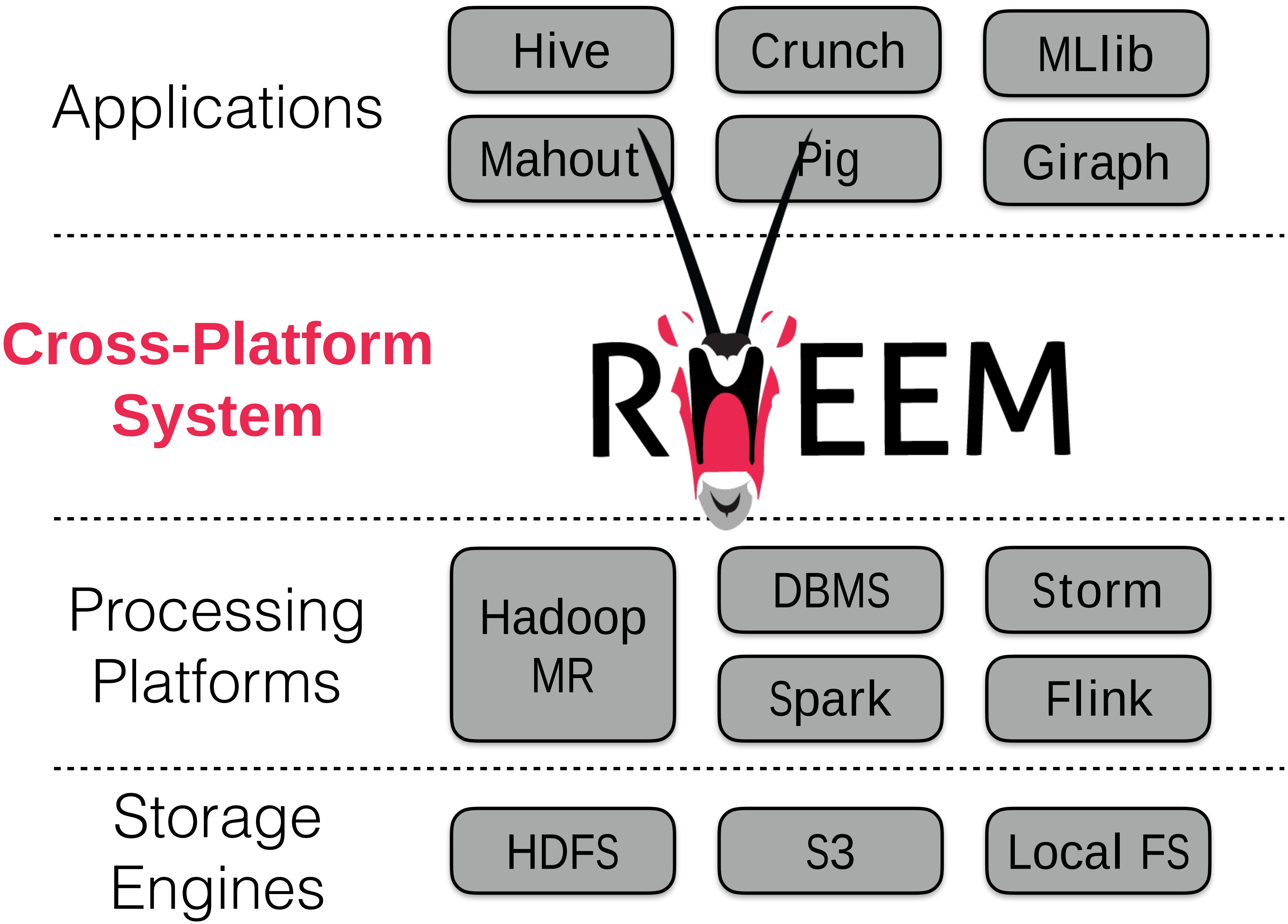
Storage Engines

HDFS

S3

Local FS

Where in the Analytics Stack?



Applications

Hive

Crunch

MLlib

Mahout

Pig

Giraph

Cross-Platform System

Processing Platforms

Hadoop MR

DBMS

Storm

Spark

Flink

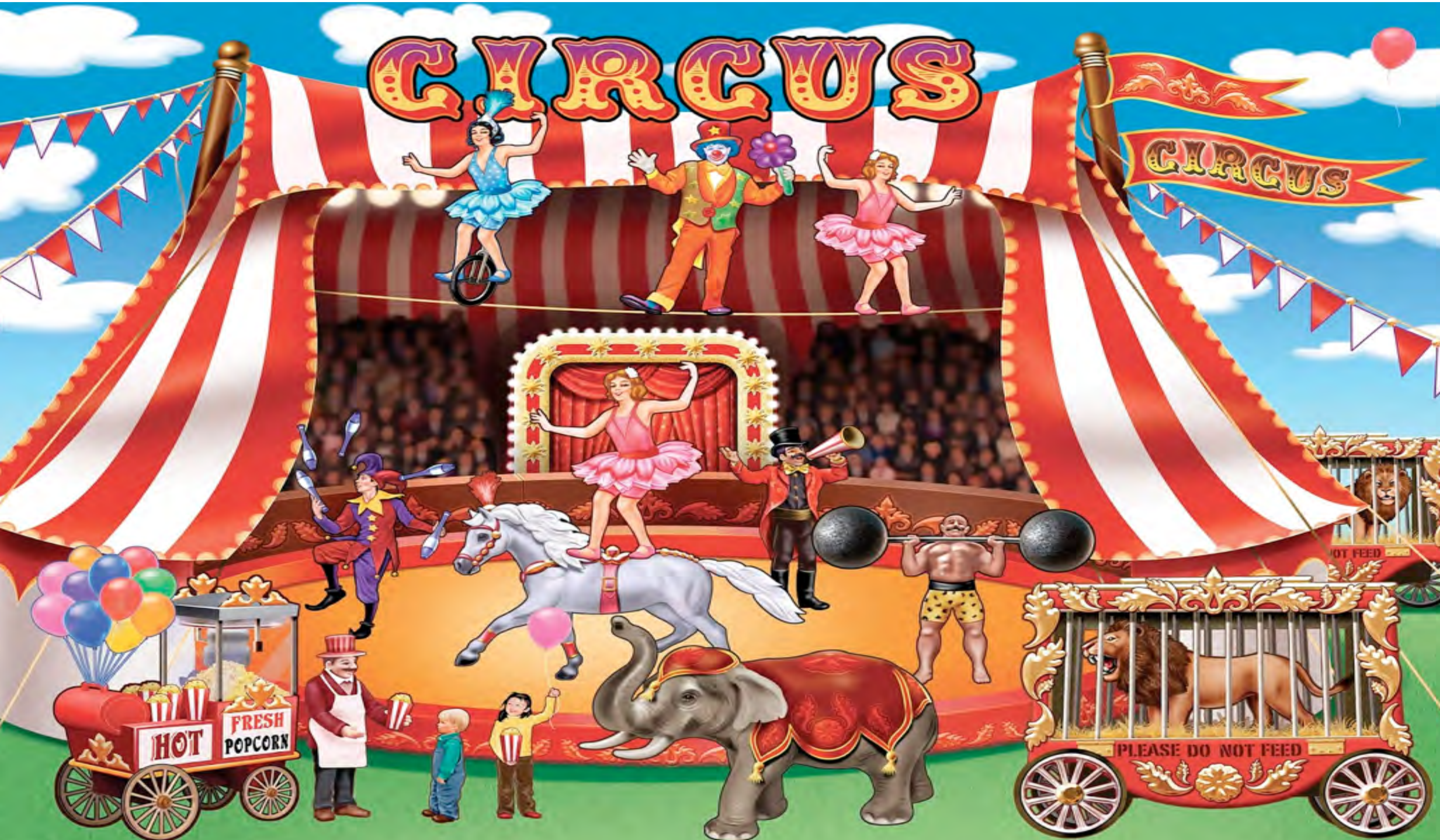
Storage Engines

HDFS

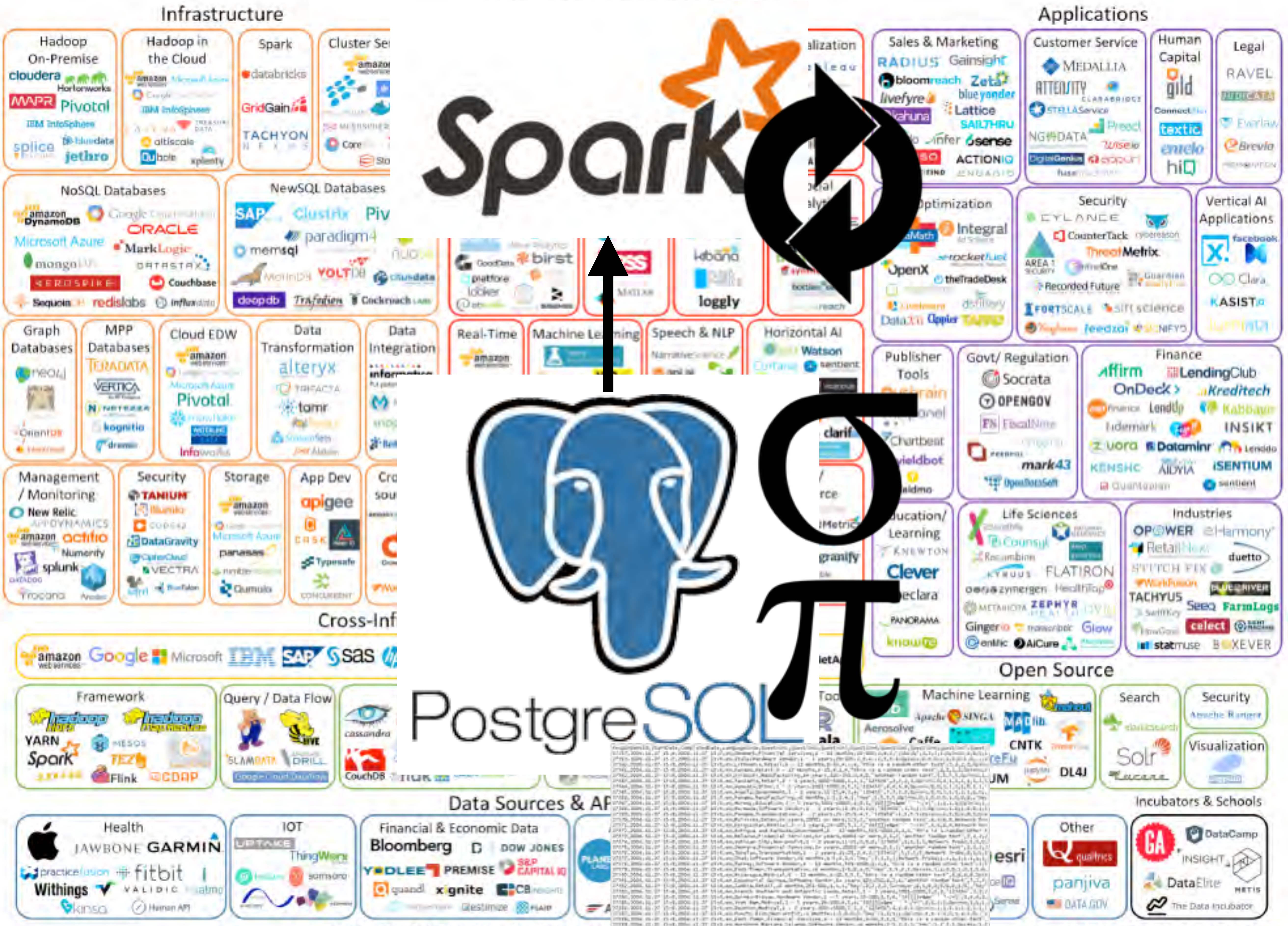
S3

Local FS

What is RHEEM for?

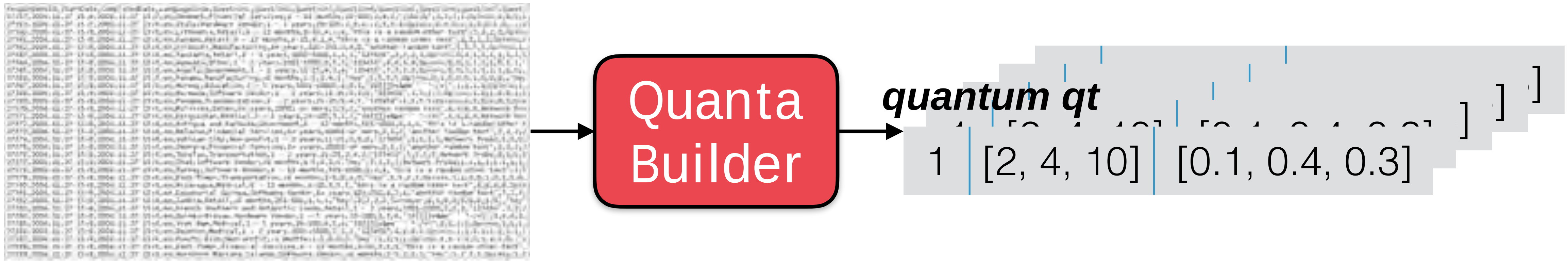


Big Data Landscape 2016



© Matt Turck (@mattturck), Jim Hao (@jimrhao), & FirstMark Capital (@firstmarkcap)

Data Model

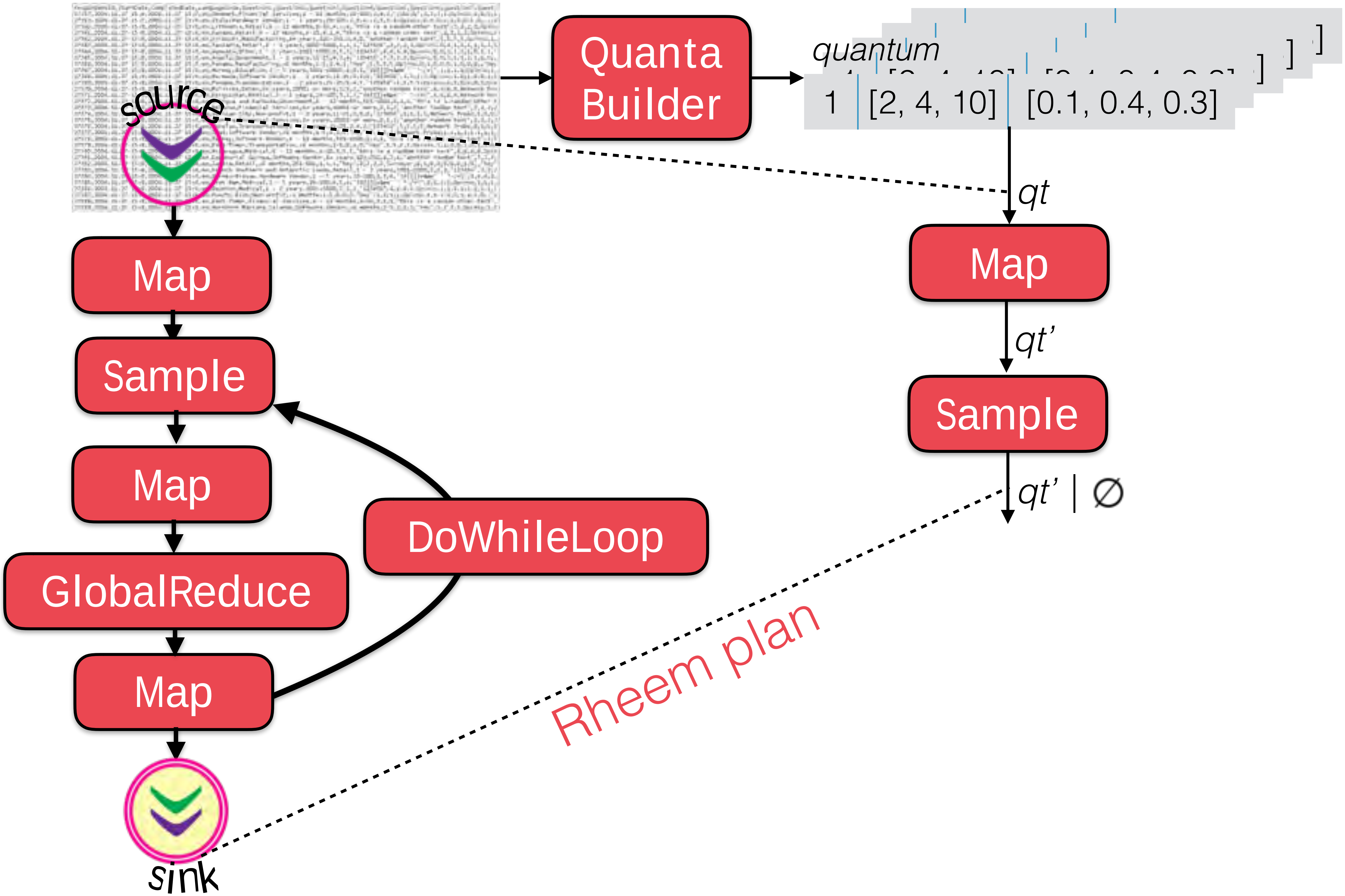


Quanta Builder

quantum qt

1	[2, 4, 10]	[0.1, 0.4, 0.3]

Processing Model



source

Quanta Builder

quantum

1 | [2, 4, 10] | [0.1, 0.4, 0.3]

qt

Map

qt'

Sample

qt' | ∅

Map

Sample

Map

GlobalReduce

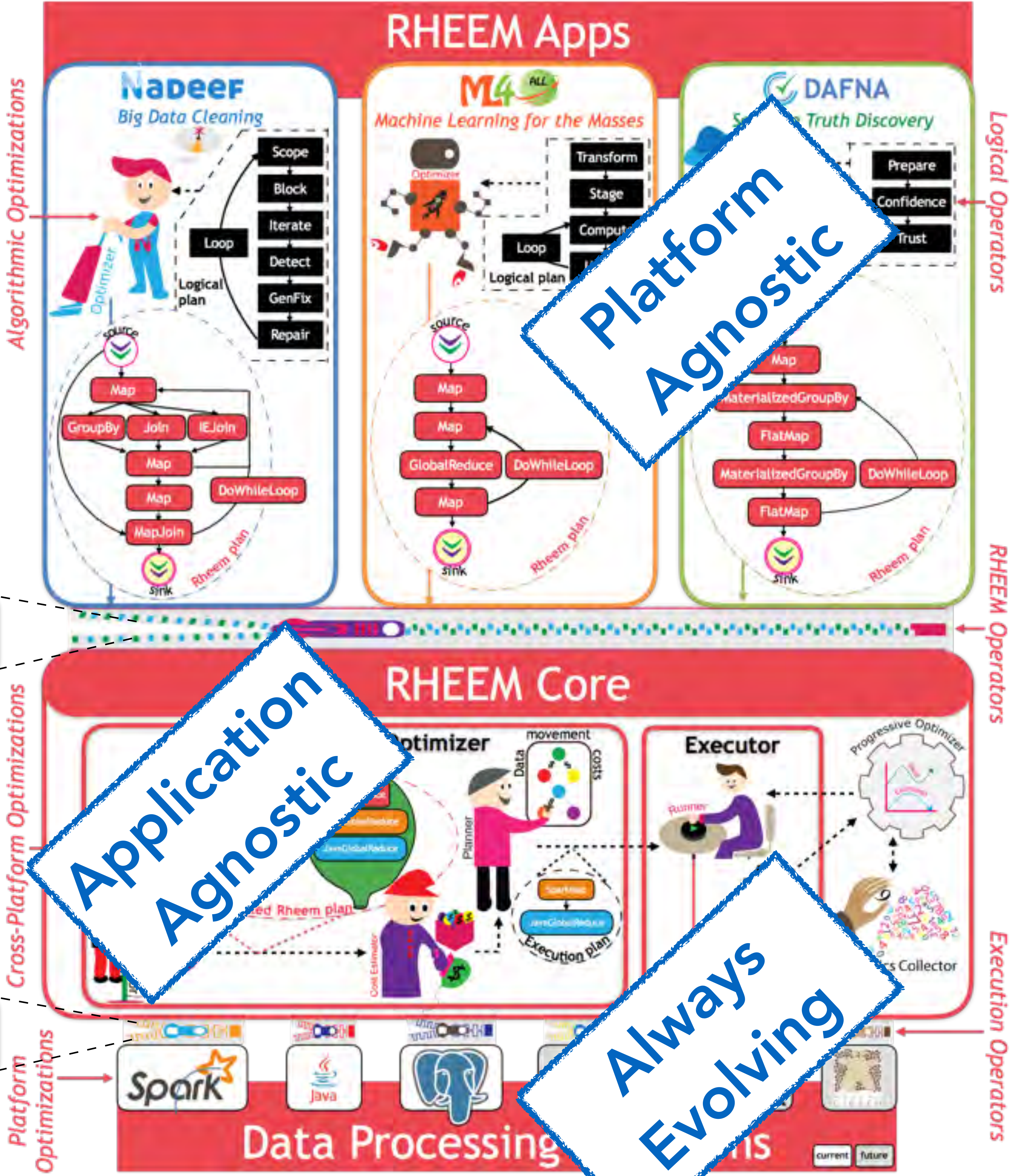
Map

DoWhileLoop

sink

Rheem plan

Three-Layer Optimization



Platform Agnostic

Application Agnostic

Always Evolving

$

$

Three-Layer Abstraction

This Tutorial

Getting Ready

- How to get Rheem
- How to setup Rheem

Hands on Rheem

- Word count
- IND discovery
- Pagerank

Demo

- ML app (ML4all)
- Extending operators

Rheem cost functions

- Execution logs
- Regression on the logs
- Cost functions

This Tutorial

Getting Ready

- How to get Rheem
- How to setup Rheem

Hands on Rheem

- Word count
- IND discovery
- Pagerank

Demo

- ML app (ML4all)
- Extending operators

Rheem cost functions

- Execution logs
- Regression on the logs
- Cost functions

Get Rheem

- **Rheem web page**
  http://da.qcri.org/rheem/
- **Rheem repository**
  https://github.com/daqcri/rheem

  ```
  $ git clone https://github.com/daqcri/rheem.git
  ```
- **Examples**
  https://github.com/sekruse/rheem-examples

  ```
  $ git clone \
          https://github.com/sekruse/rheem-examples.git
  ```
- **Useful apps**
  IntelliJ IDEA/eclipse, Git, Maven
  IPython/Jupyter with jupyter-scala kernel
- **Data**
  realworld://flash.disk/

Time to Play

Getting Ready

- How to get Rheem
- How to setup Rheem

Hands on Rheem

- Word count
- IND discovery
- Pagerank

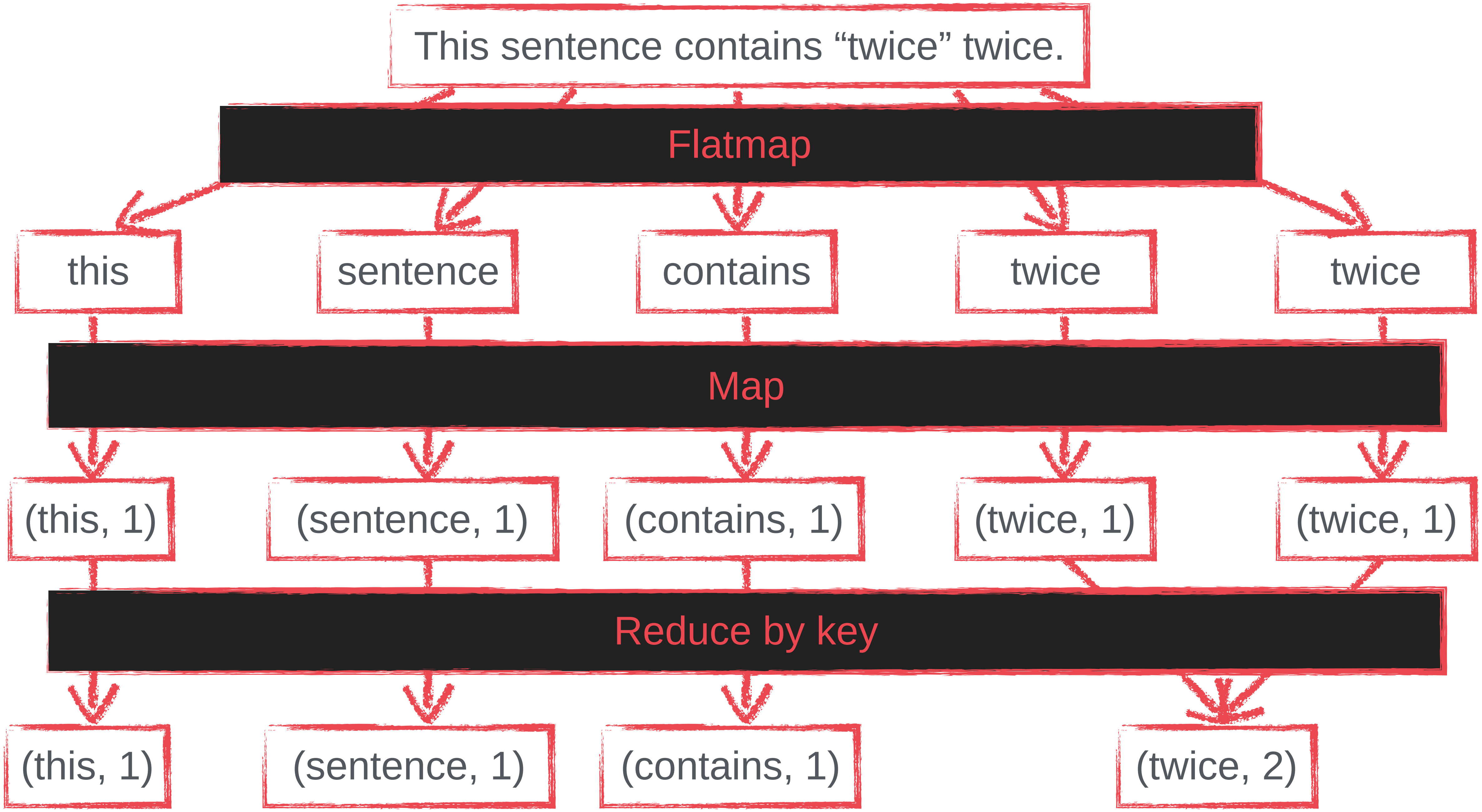
Demo

- ML app (ML4all)
- Extending operators

Rheem cost functions

- Execution logs
- Regression on the logs
- Cost functions

Wordcount



This sentence contains “twice” twice.

Flatmap

this

sentence

contains

twice

twice

Map

(this, 1)

(sentence, 1)

(contains, 1)

(twice, 1)

(twice, 1)

Reduce by key

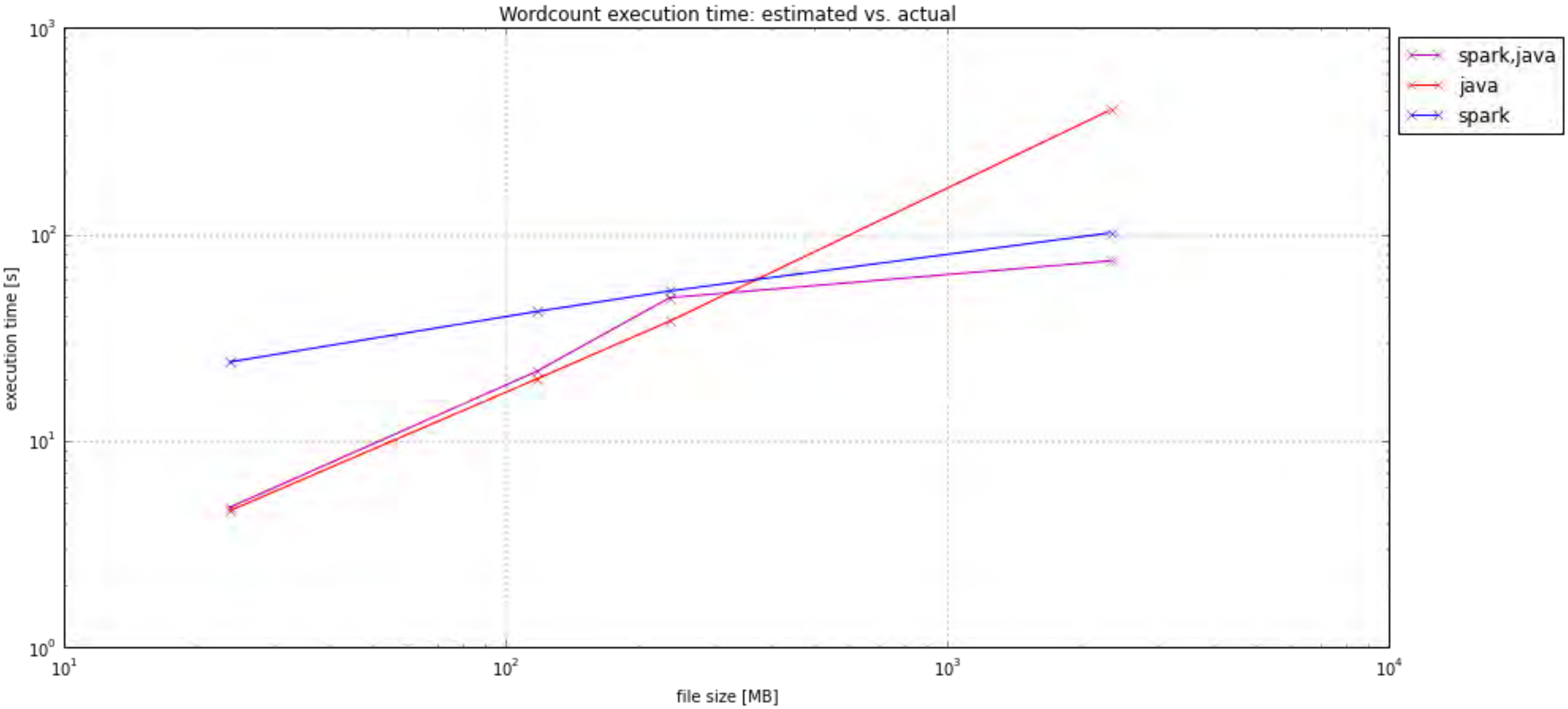
(this, 1)

(sentence, 1)

(contains, 1)

(twice, 2)

Wordcount on Rheem

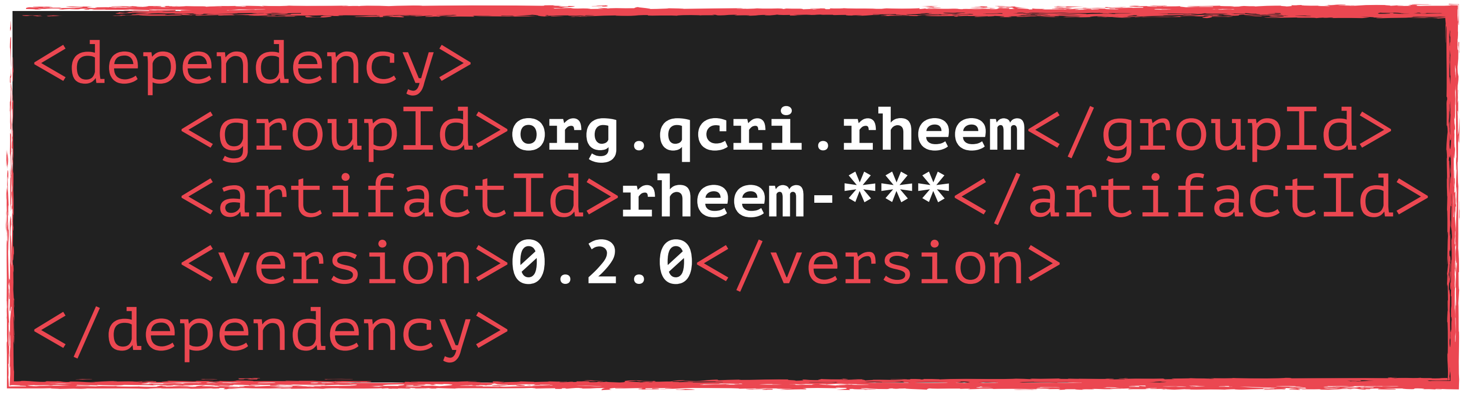


Blueprint for Rheem Apps

1. Declare Rheem dependency

1. Declare Dependencies

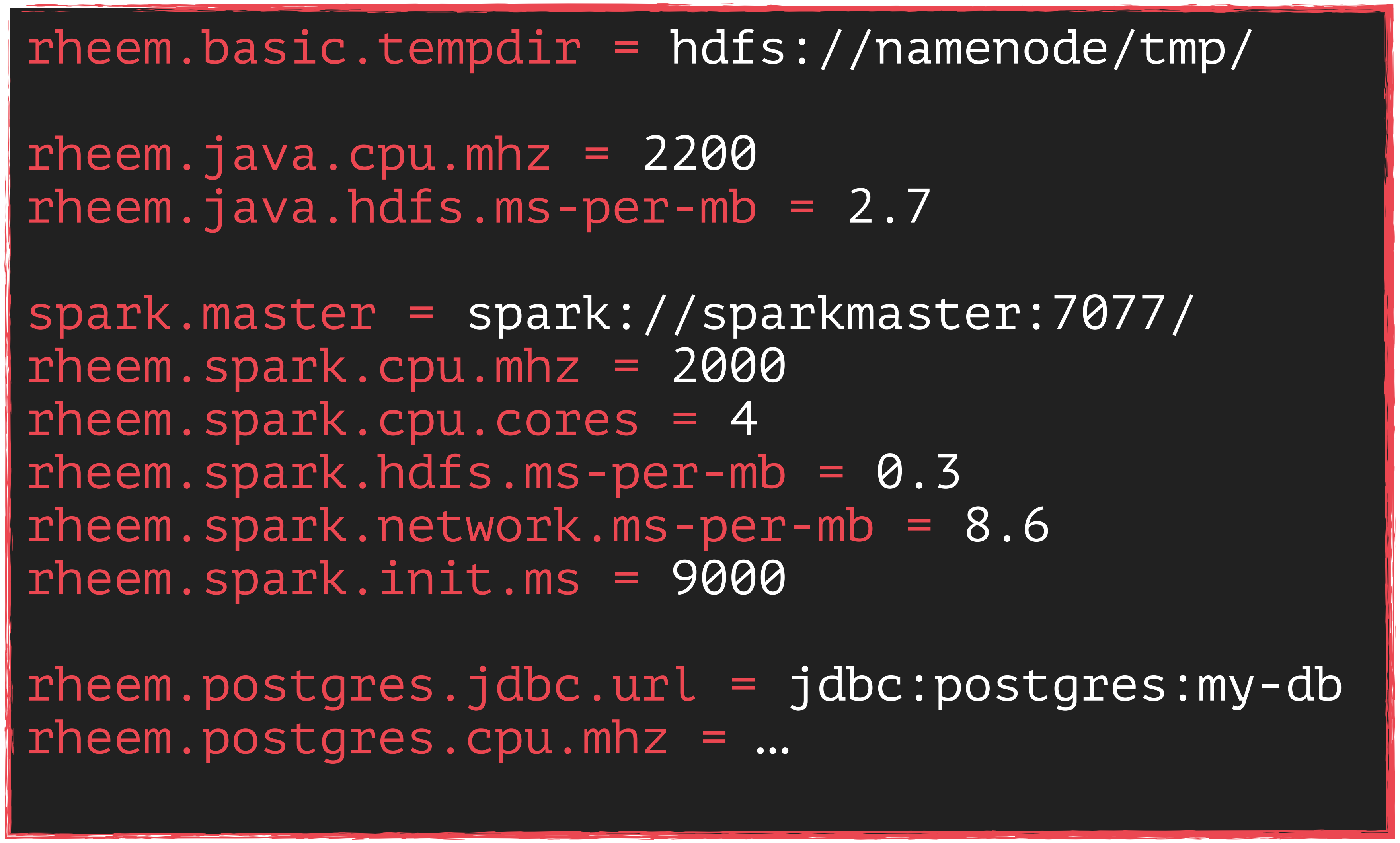
- Available in Maven Central



```
<dependency>
    <groupId>org.qcri.rheem</groupId>
    <artifactId>rheem-***</artifactId>
    <version>0.2.0</version>
</dependency>
```

  - optimizer, execution: **`rheem-core`**
  - Java and Scala API: **`rheem-api`**
  - modules: **`rheem-basic`**, **`rheem-java`**, **`rheem-spark`**, **`rheem-sqlite3`**, **`rheem-postgres`**, **`rheem-graphchi`**

2. Obtain a Configuration



```
rheem.basic.tempdir = hdfs://namenode/tmp/

rheem.java.cpu.mhz = 2200
rheem.java.hdfs.ms-per-mb = 2.7

spark.master = spark://sparkmaster:7077/
rheem.spark.cpu.mhz = 2000
rheem.spark.cpu.cores = 4
rheem.spark.hdfs.ms-per-mb = 0.3
rheem.spark.network.ms-per-mb = 8.6
rheem.spark.init.ms = 9000

rheem.postgres.jdbc.url = jdbc:postgres:my-db
rheem.postgres.cpu.mhz = …
```

2. Obtain a Configuration

- Configuration defines cost functions, advanced features, app properties
- `val configuration = new Configuration()`
  - Explicitly specify a configuration file
    `java -Drheem.configuration=url://to/my/rheem.properties` ...
  - Put a `rheem.properties` file on your classpath
  - If none applies, there are fallback values

3. Register Plugins

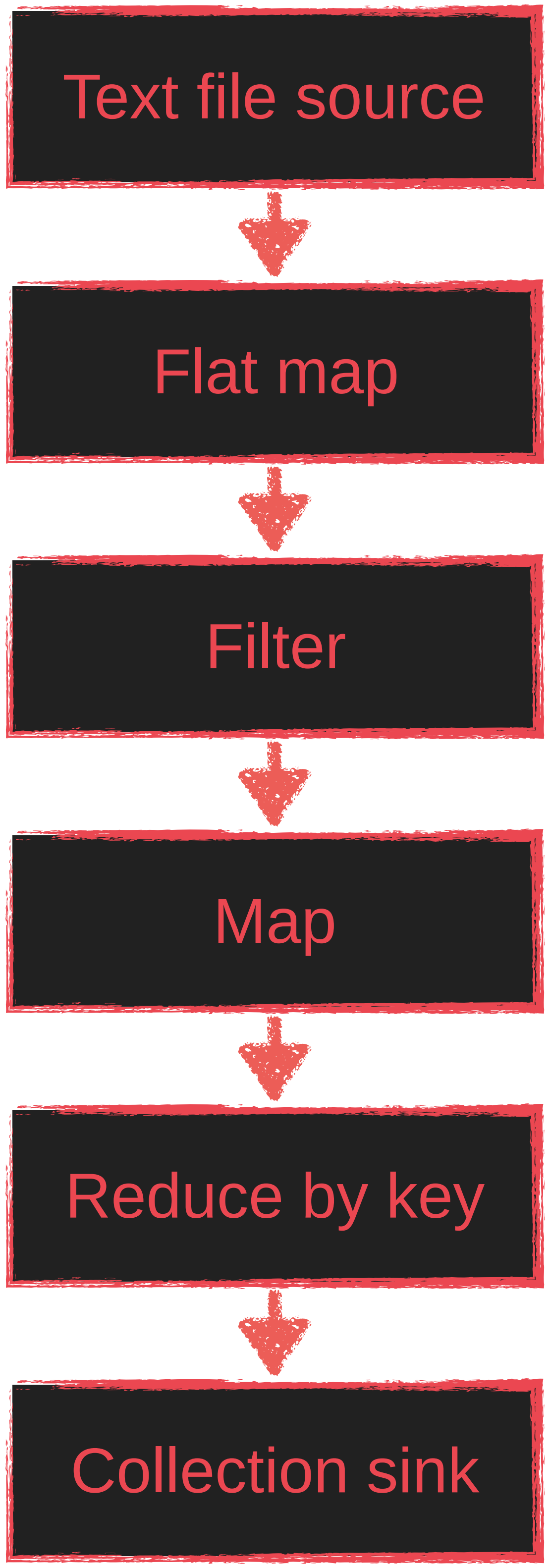
- `new RheemContext(configuration).withPlugin(...)`
- `Plugins` provide execution platforms and/or operators
- Available plugins:

4. Build a Rheem Plan (I)

Start with `PlanBuilder` and chain operations.

```
val planBuilder =
 new PlanBuilder(rheemContext)
  .withJobName(s"WordCount ($url)")
  .withUdfJarsOf(this.getClass)

val wordCounts = planBuilder
  .readTextFile(url)
  .flatMap(…, selectivity = …)
  .filter(…, selectivity = …)
  .map(…)
  .reduceByKey(…)
  .collect()
```



4. Build a Rheem Plan (II)

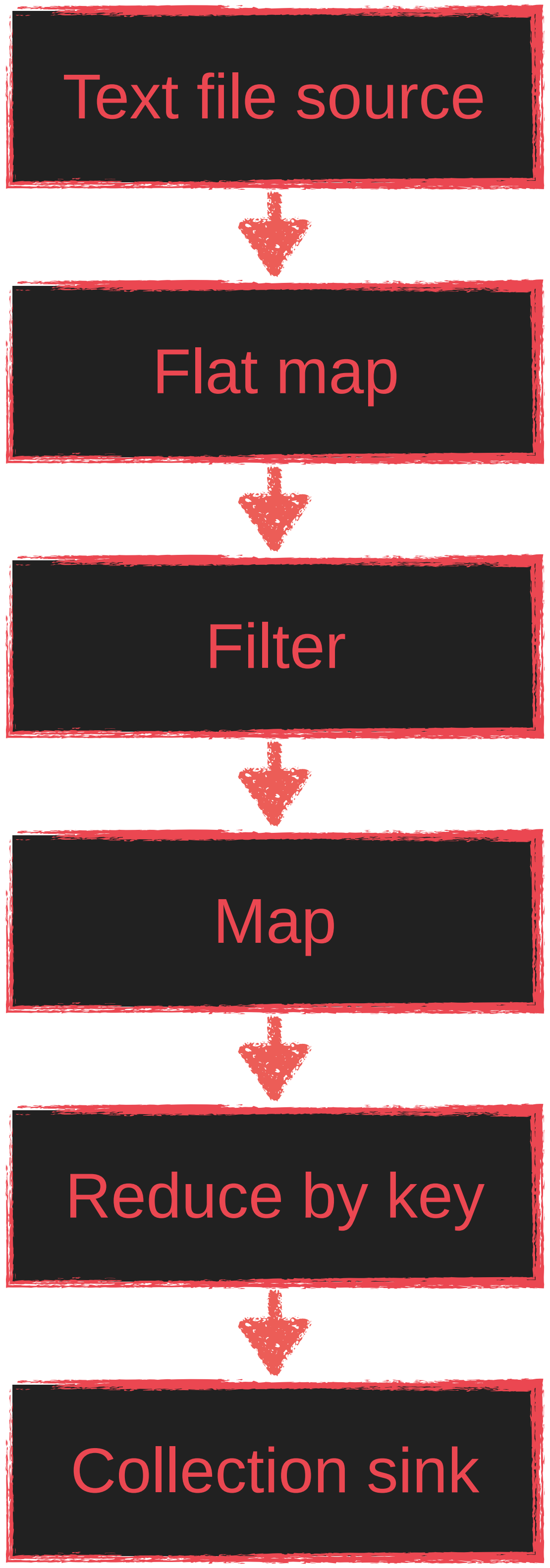
Rheem supports loops, which is important for machine learning algorithms:

```
val myResult = …
 .repeat(n, i => i.map(…))
 …
 .collect()
```

Data flows can be joined in a flexible manner:

```
val sizeDataset = myDataset.count

val result = myDataset
 .map(…).withBroadcast(
   sizeDataset, “size”
 ).collect()
```



Text file source

Flat map

Filter

Map

Reduce by key

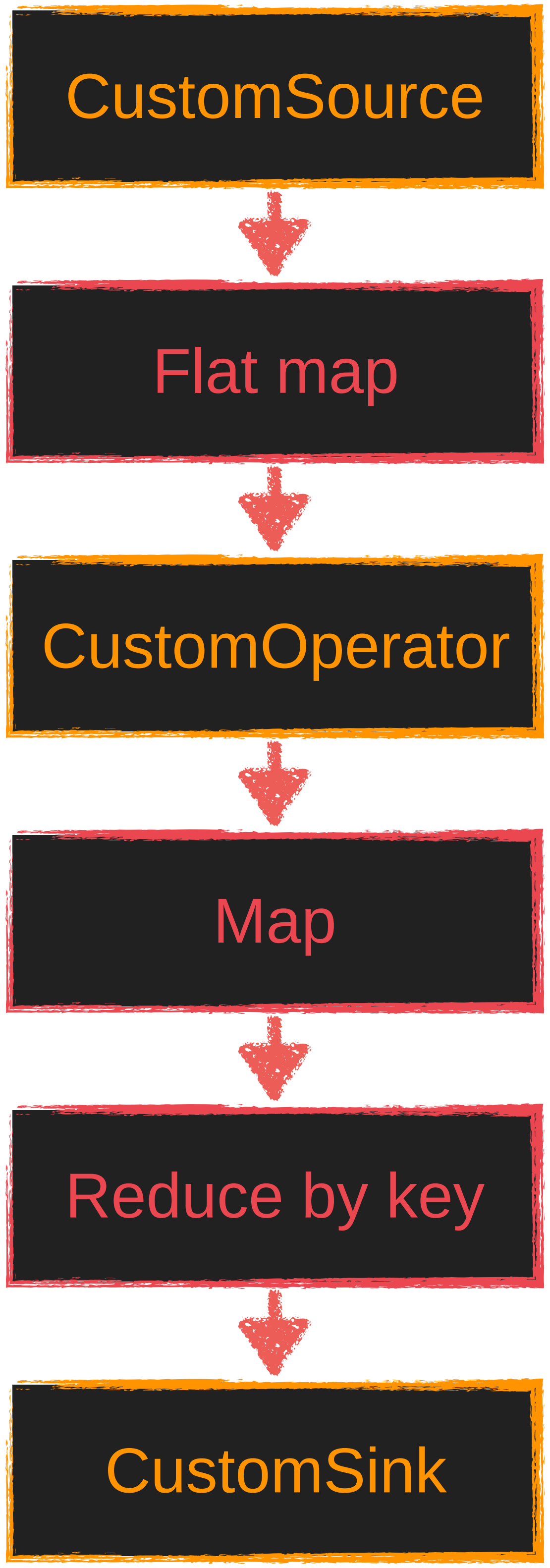
Collection sink

4. Build a Rheem Plan (III)

API is just syntactic sugar - use your own operators!

```
planBuilder.load(new CustomSource())
  .flatmap(…)
  .customOperator(new CustomOperator())
  .map(…)

  .reduceByKey(…)
  .customOperator(new CustomSink())
```

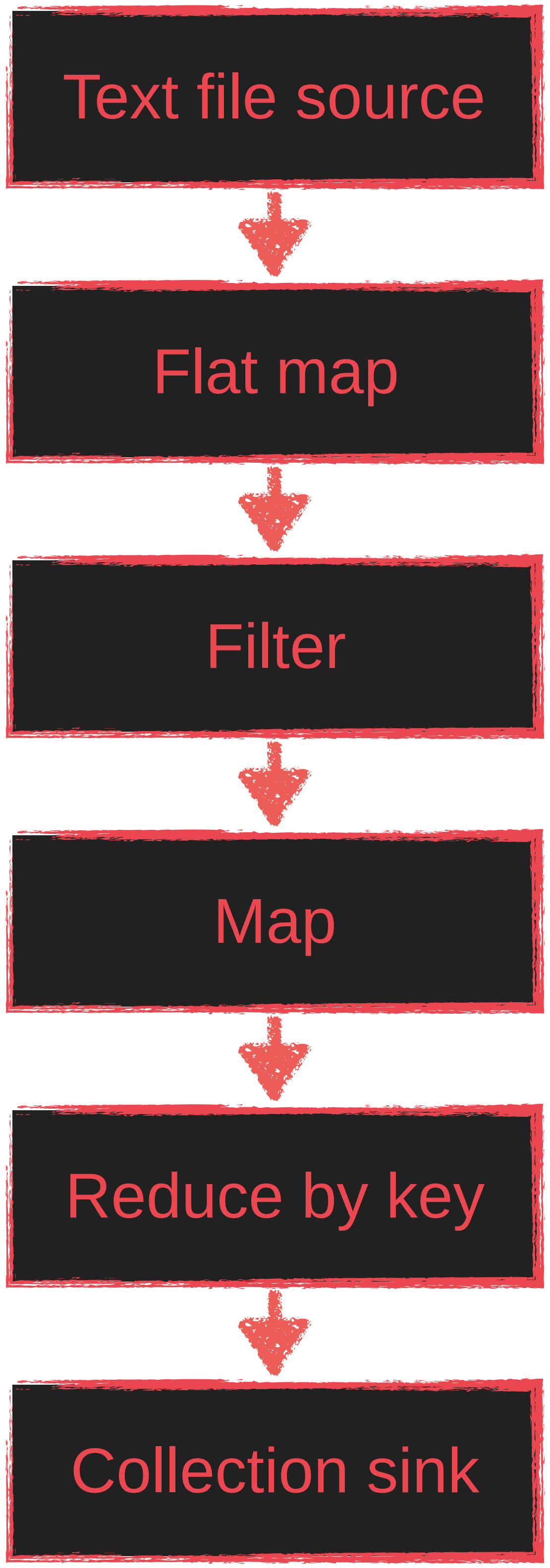


5. Trigger Execution

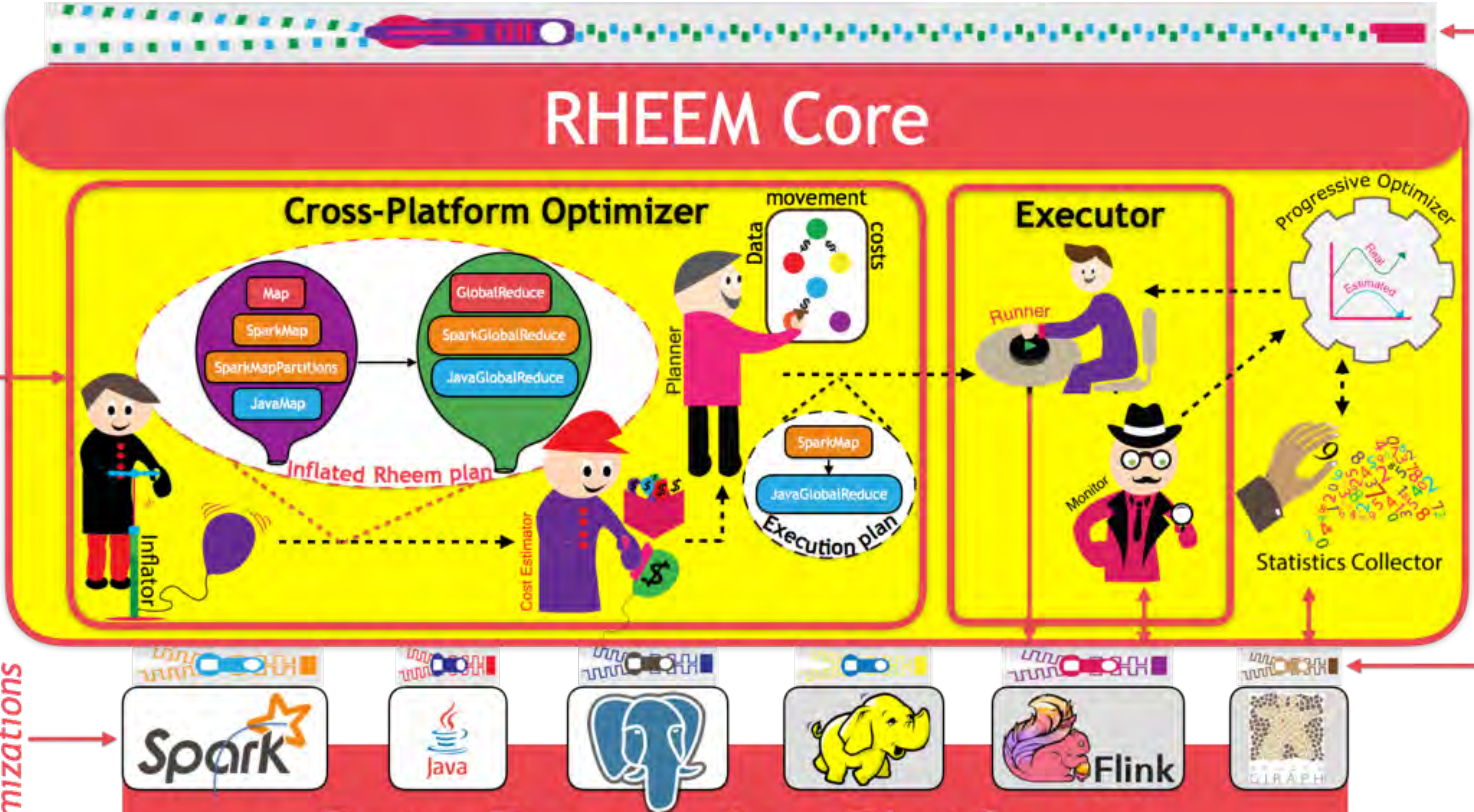
- Design choice: creating a sink triggers execution
  - Rheem allows for multiple sinks → stay tuned

- Available sinks
  - ….`collect()`: fetch dataset as JVM-based collection
  - ….`writeAsTextFile(...)`: format & write dataset to a text file

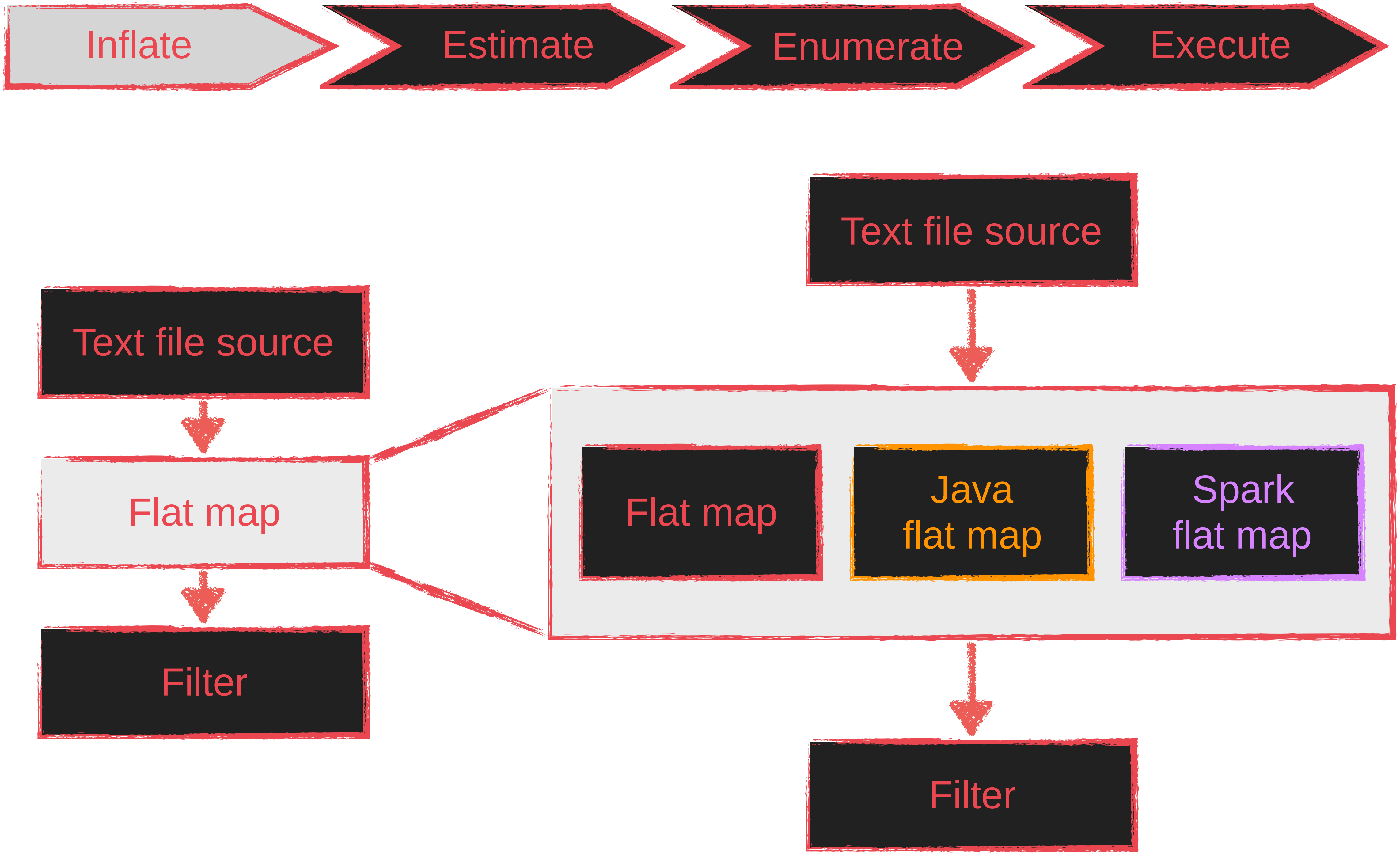
- Note: Only when execution is triggered, Rheem start its optimization, let alone execution.



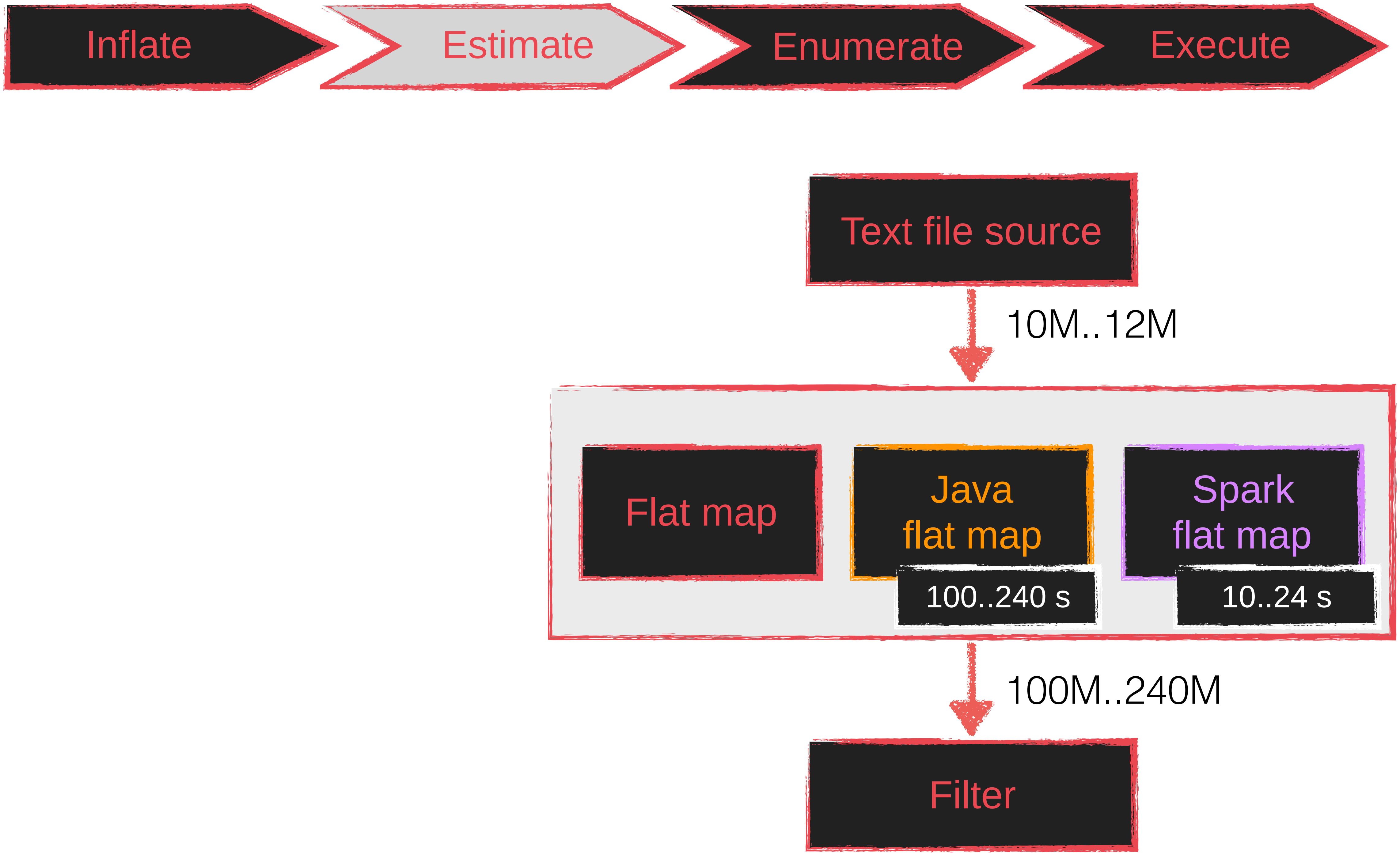
6. Let Rheem do the Rest



6. Let Rheem do the Rest



6. Let Rheem do the Rest



Inflate

Estimate

Enumerate

Execute

Text file source

10M..12M

Flat map

Java flat map

100..240 s

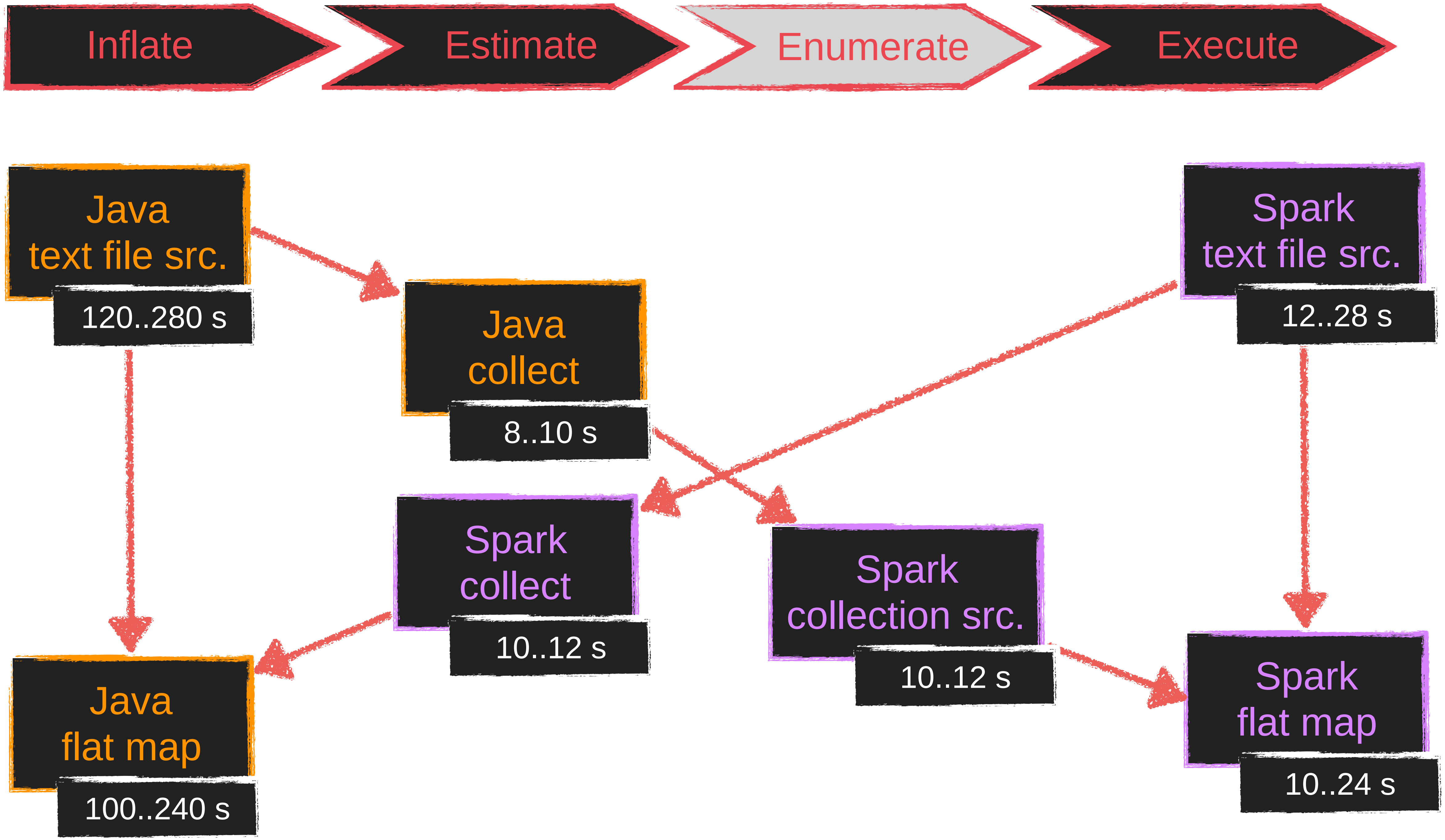
Spark flat map

10..24 s

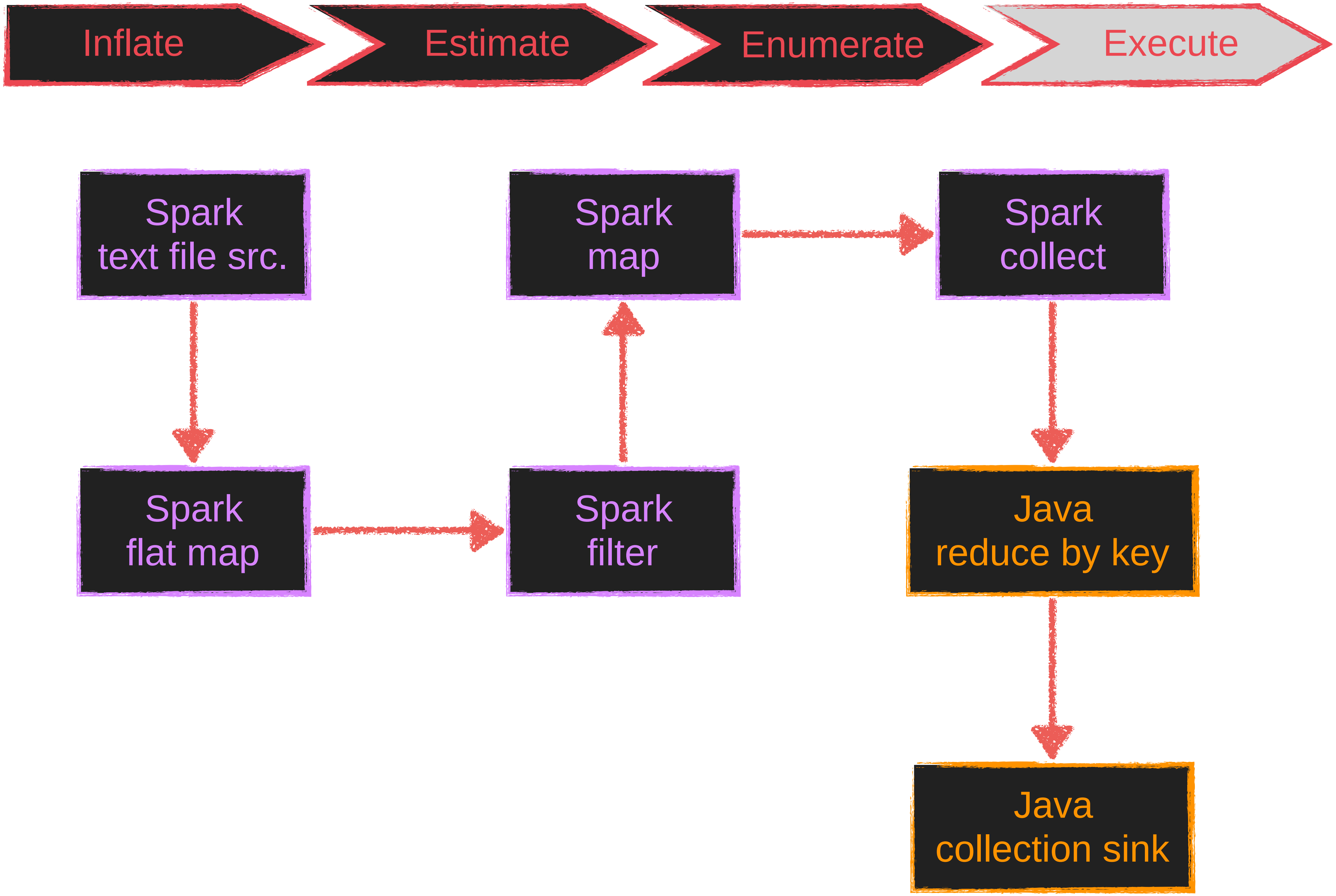
100M..240M

Filter

6. Let Rheem do the Rest



6. Let Rheem do the Rest



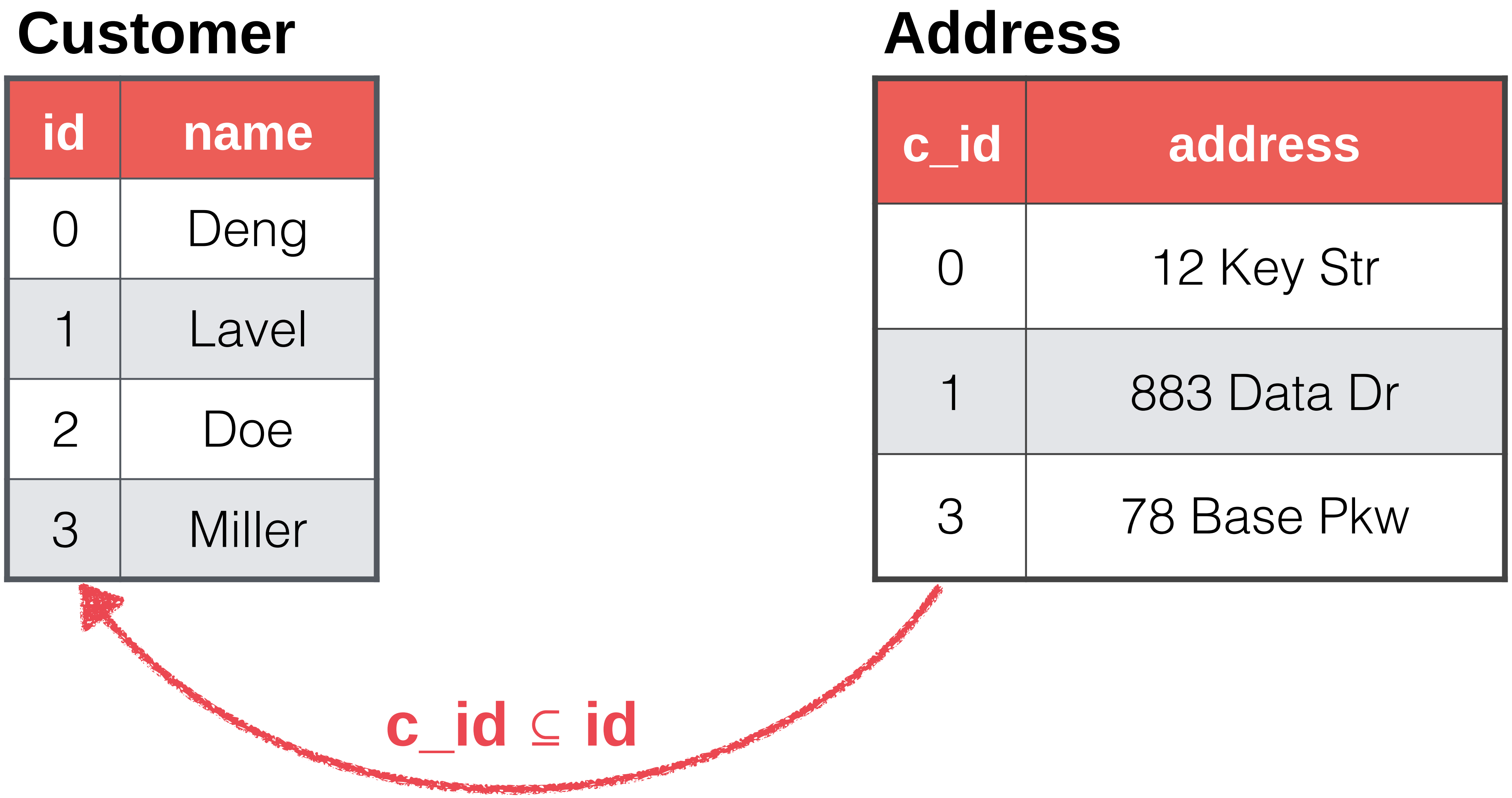
Task: Wordcount’

- Reduce number of Rheem operators.

- What are possible implications of doing so?
  - w.r.t. performance?
  - w.r.t. optimization hints?
  - w.r.t. maintainability?
  - w.r.t. Rheem’s optimization pipeline?

IND Discovery

Detect column pairs in a database, such that all values of the one are included in the other.



Customer

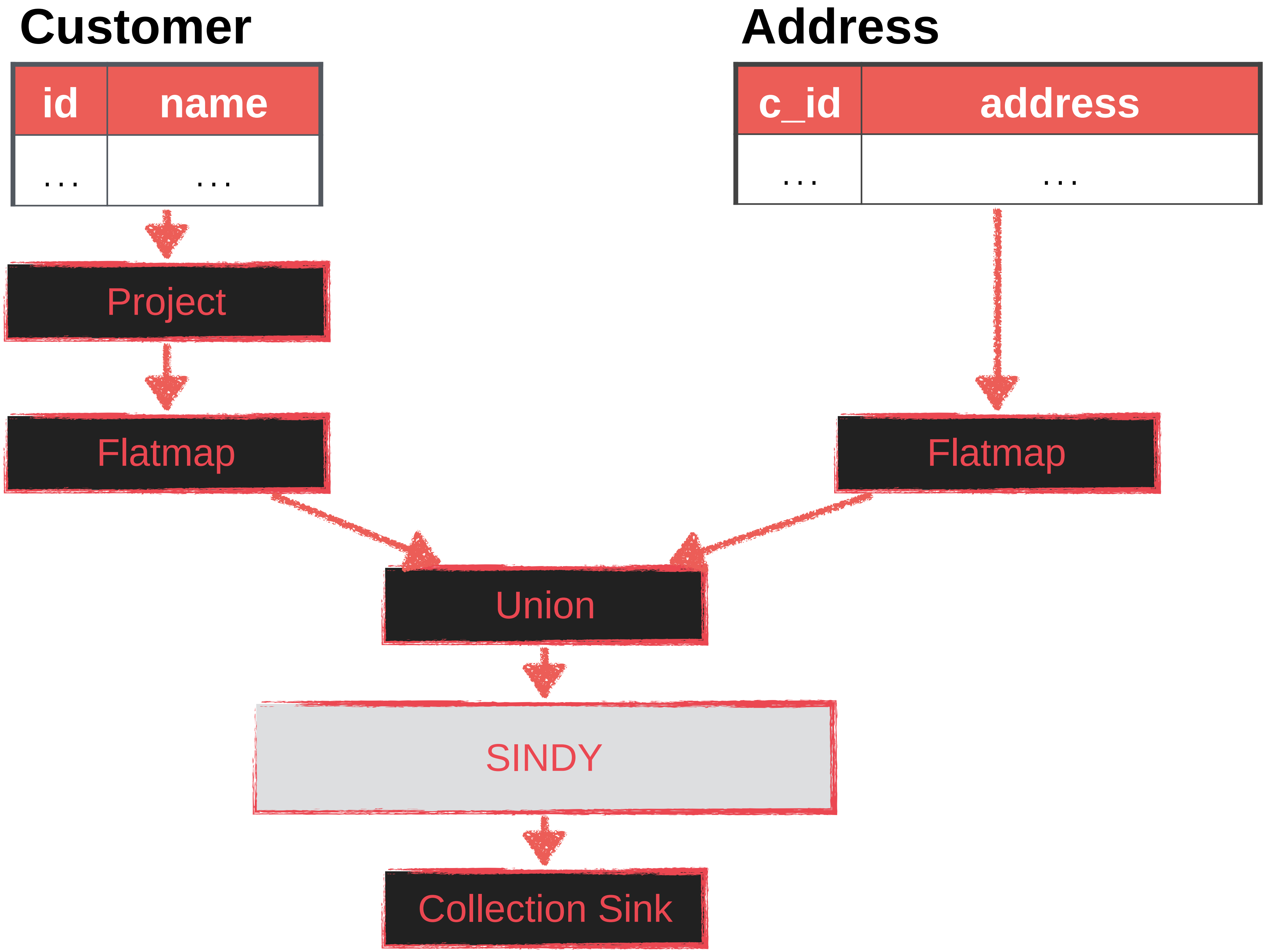
id	name
0	Deng
1	Lavel
2	Doe
3	Miller

Address

c_id	address
0	12 Key Str
1	883 Data Dr
3	78 Base Pkw

c_id ⊆ id

IND Discovery



Customer

id	name
...	...

Address

c_id	address
...	...

Project

Flatmap

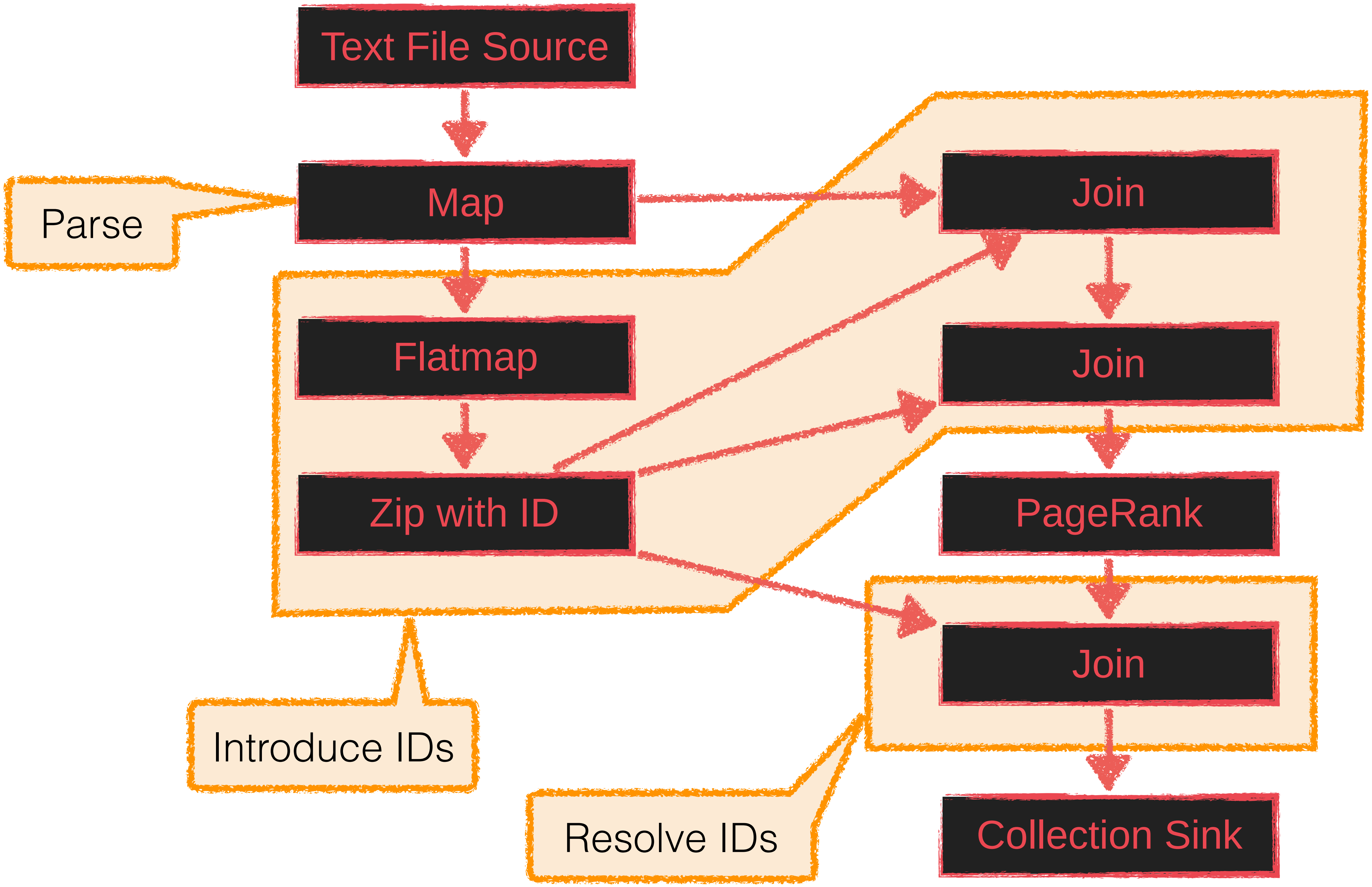
Flatmap

Union

SINDY

Collection Sink

RDF PageRank



Text File Source

Map

Parse

Flatmap

Zip with ID

Introduce IDs

Join

Join

PageRank

Join

Resolve IDs

Collection Sink

Task: Tune PageRank

- Entities URLs all conform to the pattern
  `<http://dbpedia.org/resource/…>`
  → eliminate that redundancy
- `<http://dbpedia.org/resource/Category:…>`
  are no real entities
  → remove them
- boost entities having a lot of outgoing links
  → make graph undirected

Advanced App

Getting Ready

- How to get Rheem
- How to setup Rheem

Hands on Rheem

- Word count
- IND discovery
- Pagerank

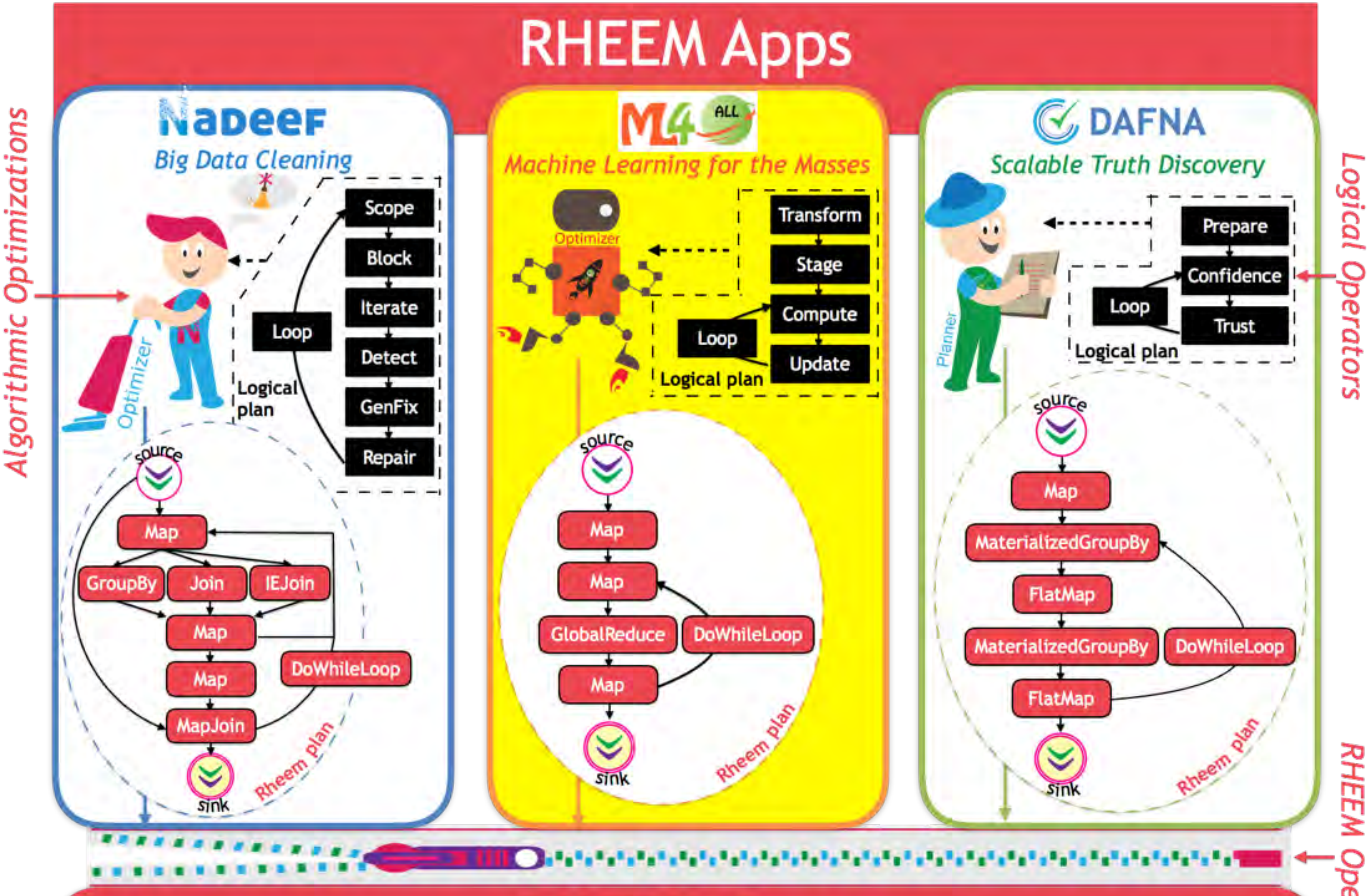
Demo

- ML app (ML4all)
- Extending operators

Rheem cost functions

- Execution logs
- Regression on the logs
- Cost functions

ML4all: ML on top of Rheem



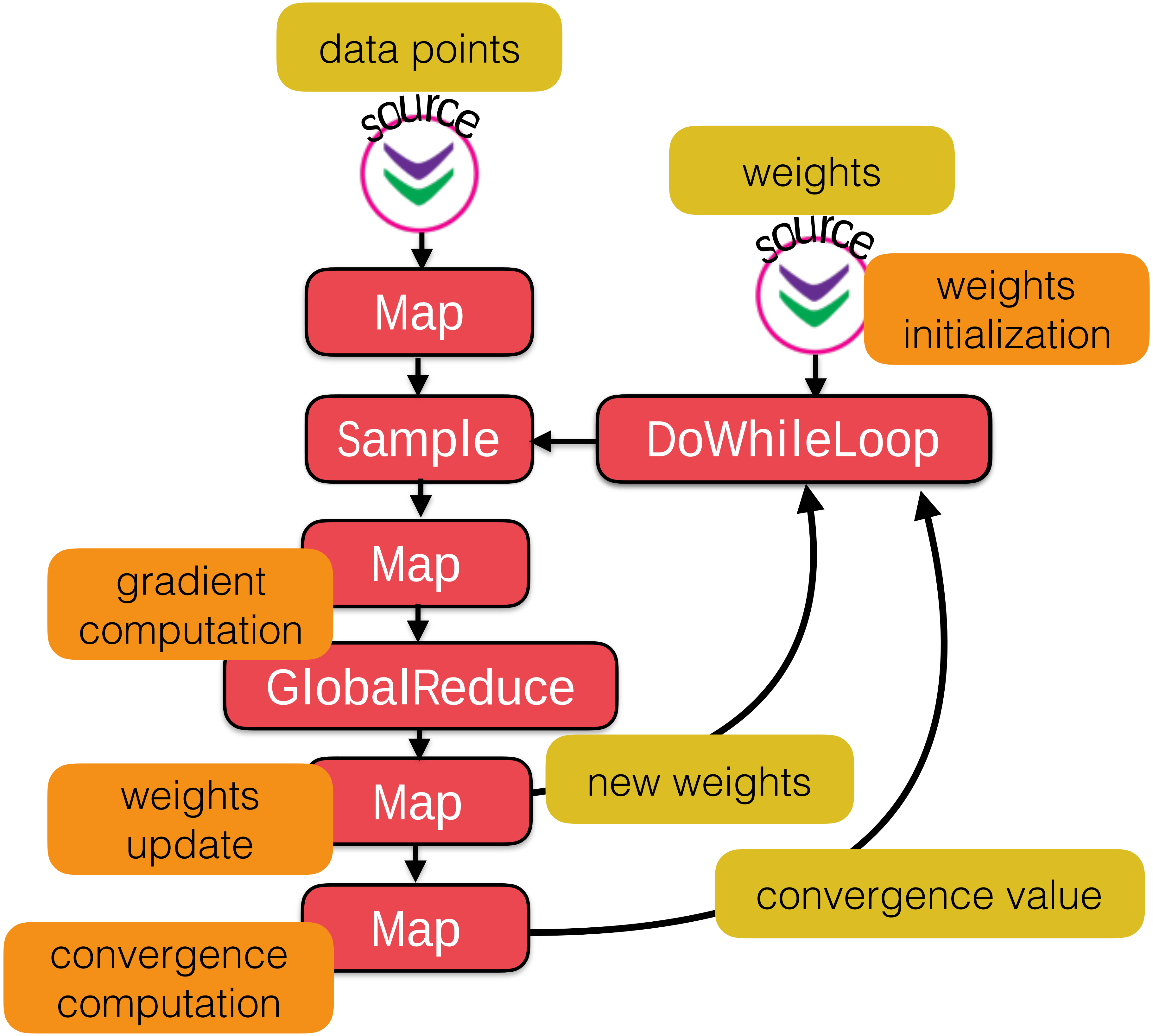
Gradient Descent

```
initialize $w^0$
while !converged {
    grad = $\Sigma_i$ gradient($f_i$(w)), for all i in D
    $w^{k+1}$:= $w^k$ + $\alpha_k$ * 1/n * grad
    converged := $||w^{k+1} - w^{k+1}|| < 0.001$
    k := k+1
}
```

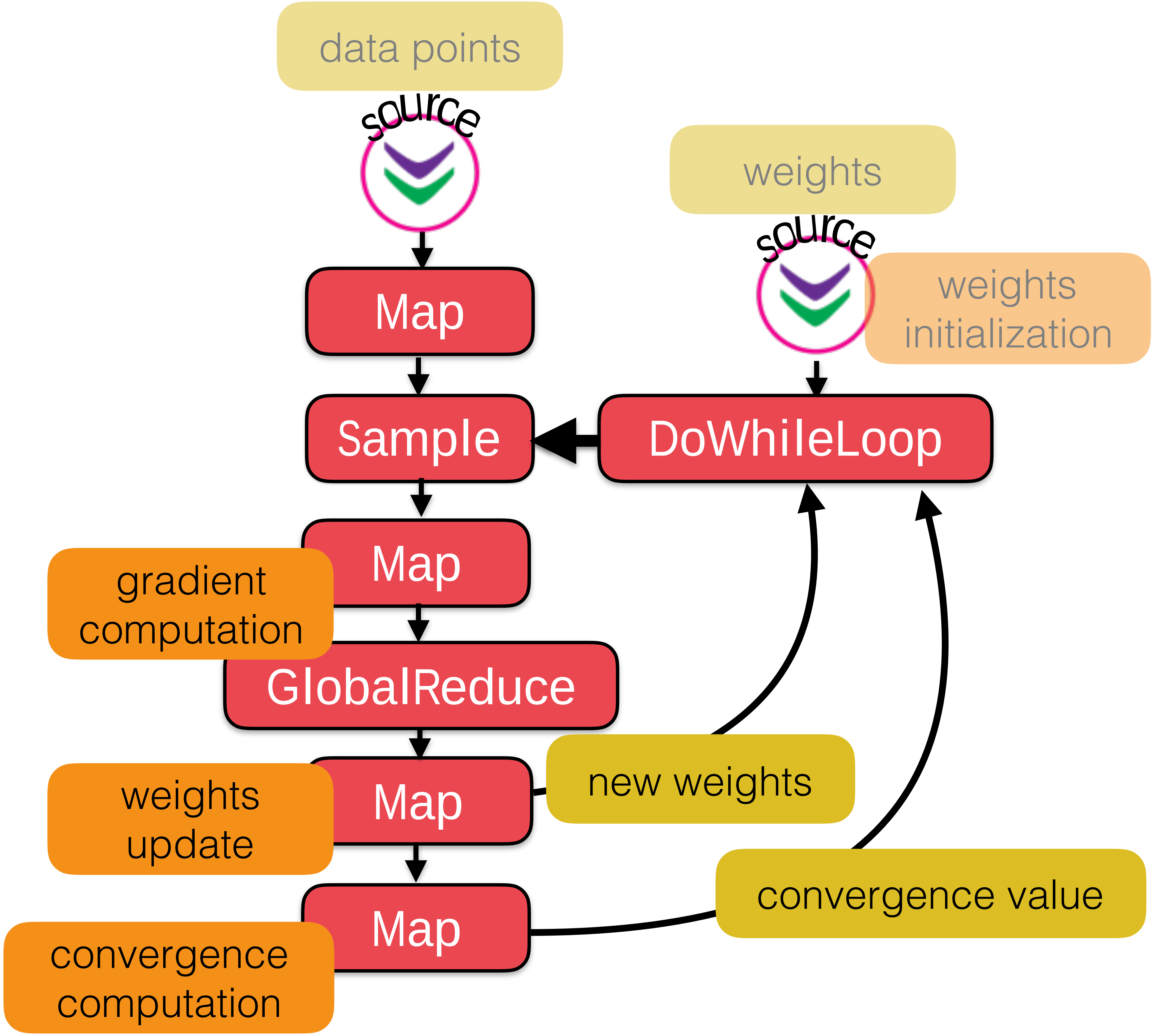
Stochastic Gradient Decent (SGD)

```
initialize $w^0$                                   weights initialization
while !converged {
        j := sample from D
        grad = gradient($f_j(w)$)                  gradient computation
        $w^{k+1}$ := $w^k + \alpha_k$ * grad       weights update
        converged := $||w^{k+1} - w^{k+1}|| < 0.001$   convergence
        k := k+1                                   computation
}
```

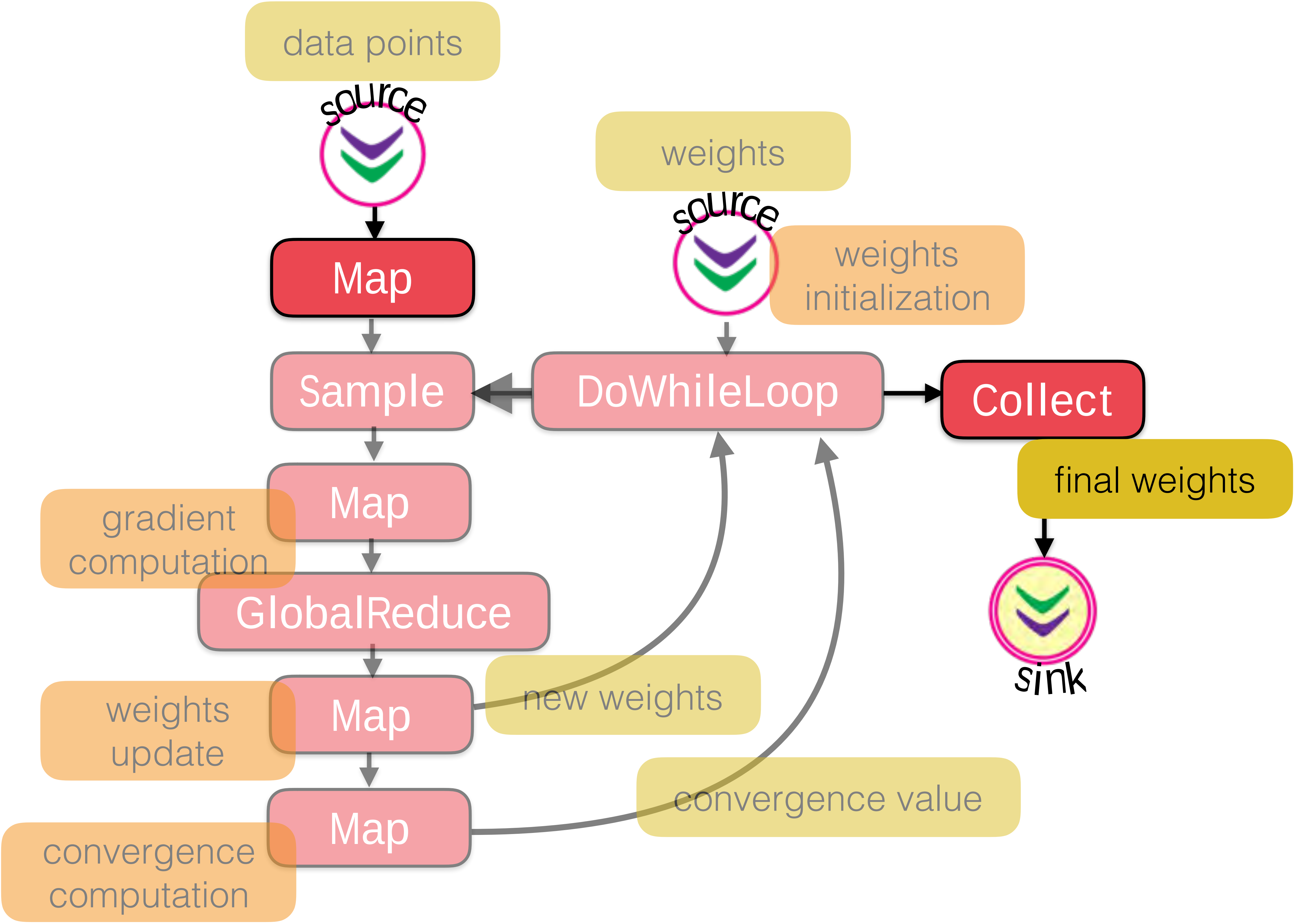
SGD Plan in Rheem



SGD Plan in Rheem



SGD Plan in Rheem



Extending Rheem with new operators

- Create core ***Rheem*** (***platform-agnostic***) ***operator***
  - Consider adding a cardinality estimator
- Create ***execution*** (***platform-specific***) ***operator***
  - Consider adding a cost function
- Create **mappings** from the Rheem operator to the execution operator

Bootstrapping Rheem

Getting Ready

- How to get Rheem
- How to setup Rheem

Hands on Rheem

- Word count
- IND discovery
- Pagerank

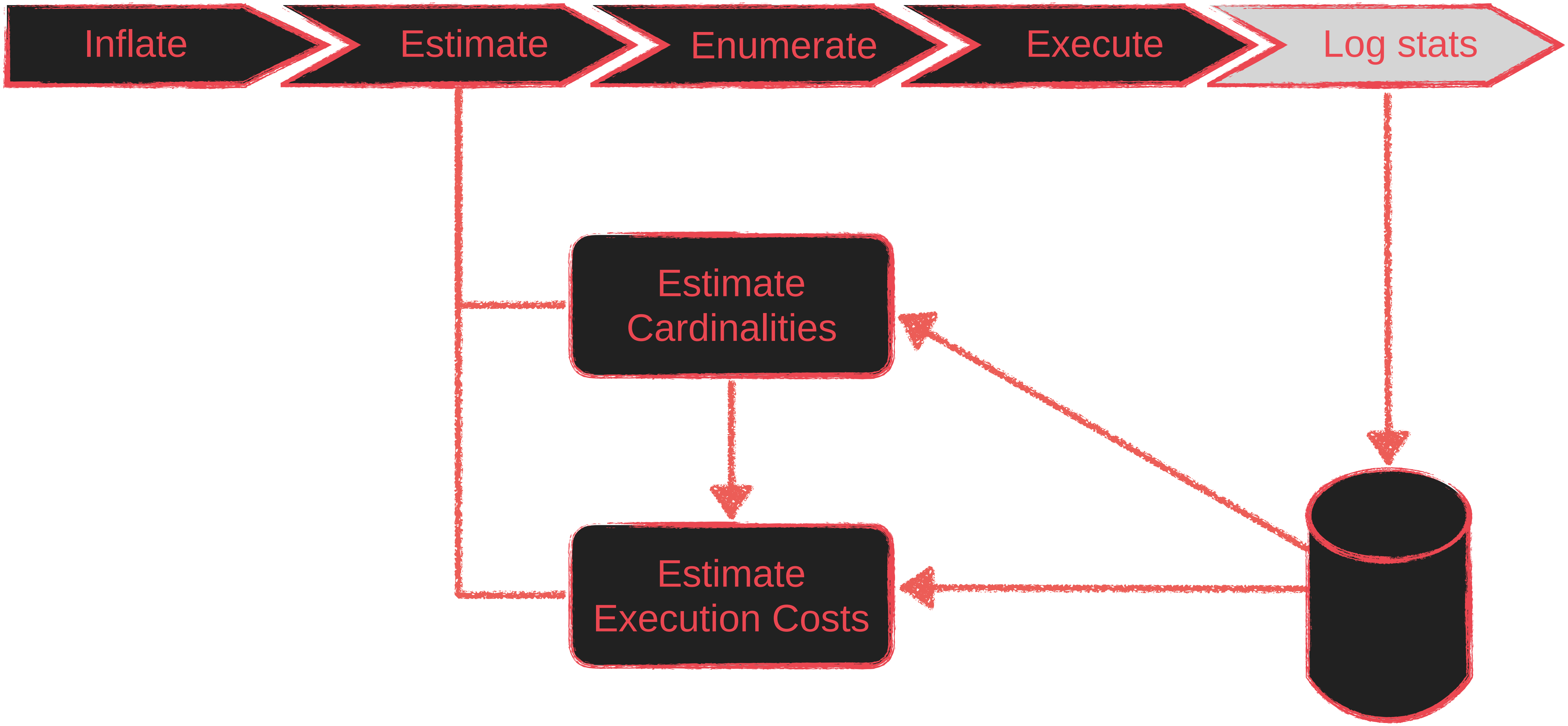
Demo

- ML app (ML4all)
- Extending operators

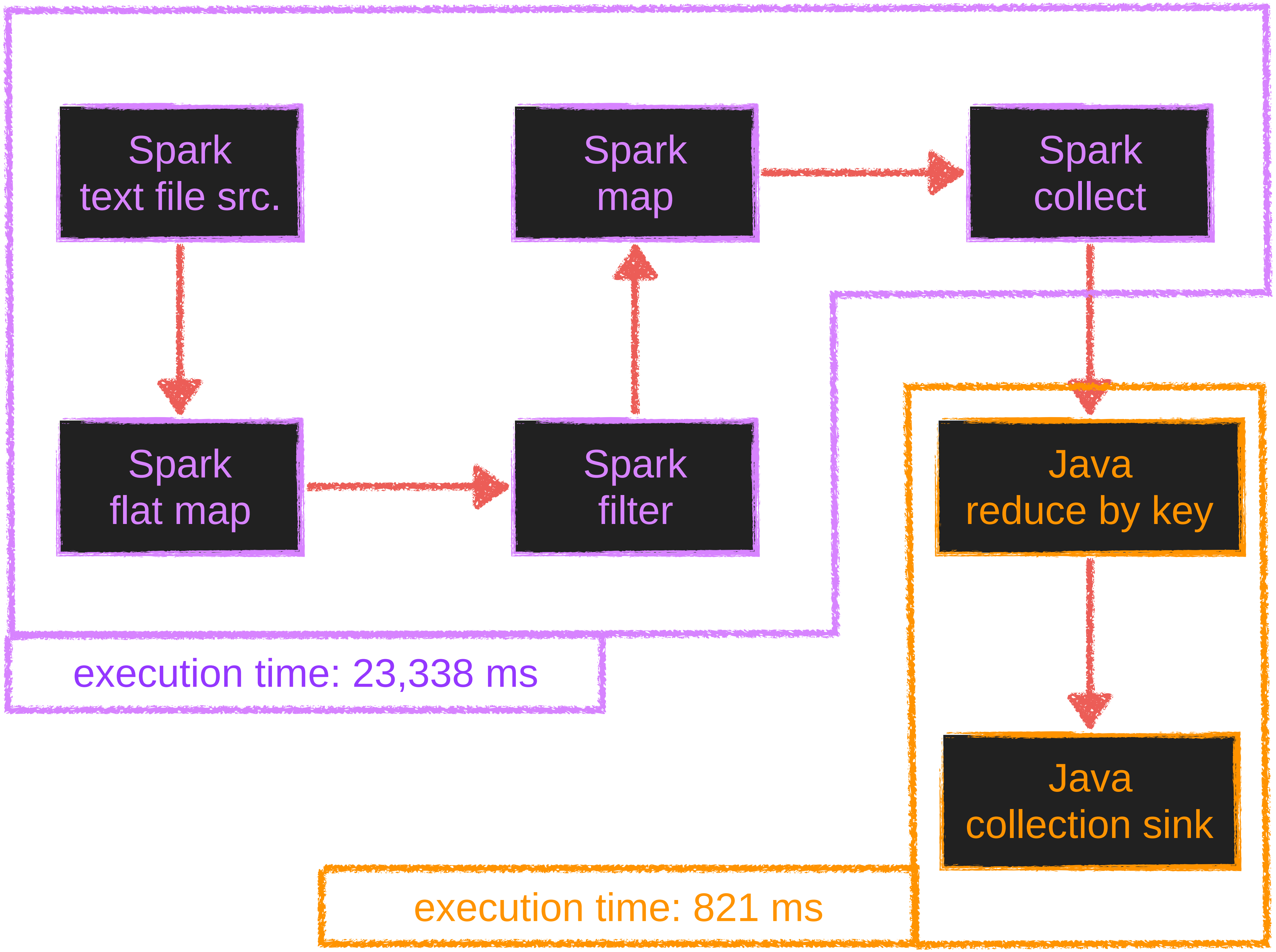
Rheem cost functions

- Execution logs
- Regression on the logs
- Cost functions

Rheem Workflow



Lazy Execution



Spark
text file src.

Spark
flat map

Spark
filter

Spark
map

Spark
collect

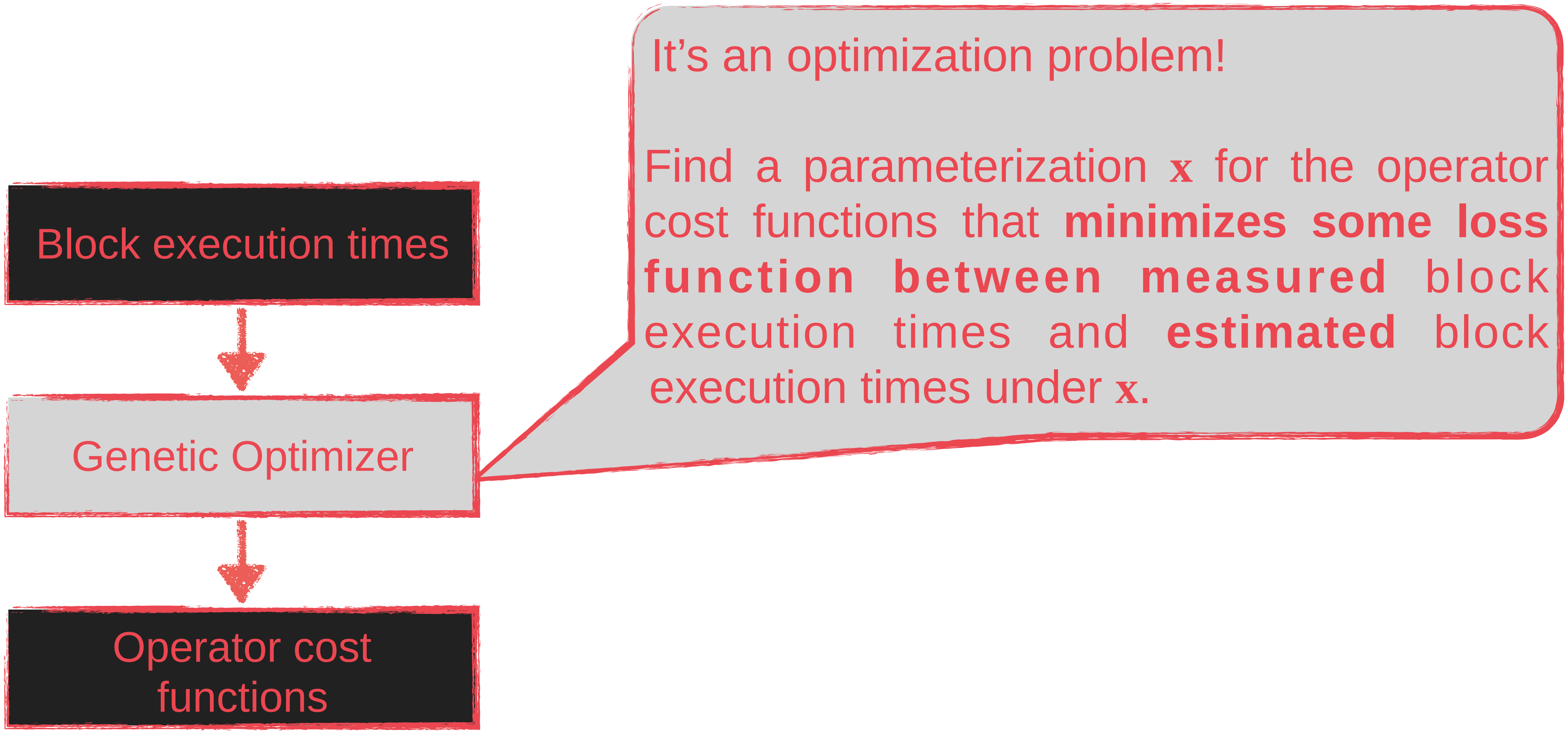
Java
reduce by key

Java
collection sink

execution time: 23,338 ms

execution time: 821 ms

Resolving Cost Functions



Block execution times

Genetic Optimizer

Operator cost functions

It’s an optimization problem!

Find a parameterization **x** for the operator cost functions that **minimizes some loss function between measured** block execution times and **estimated** block execution times under **x**.

Applying Genetic Optimization

Collecting execution data

- Rheem must keep track properly of when is what executed
  → operators are self-descriptive in that respect
- Get cardinality estimates as accurate as possible
  → Rheem monitors cardinalities and updates all estimates
- Gather sufficient, variant execution data

Complement execution data

- Provide models for the cost functions
- Account for heavy-weight UDFs

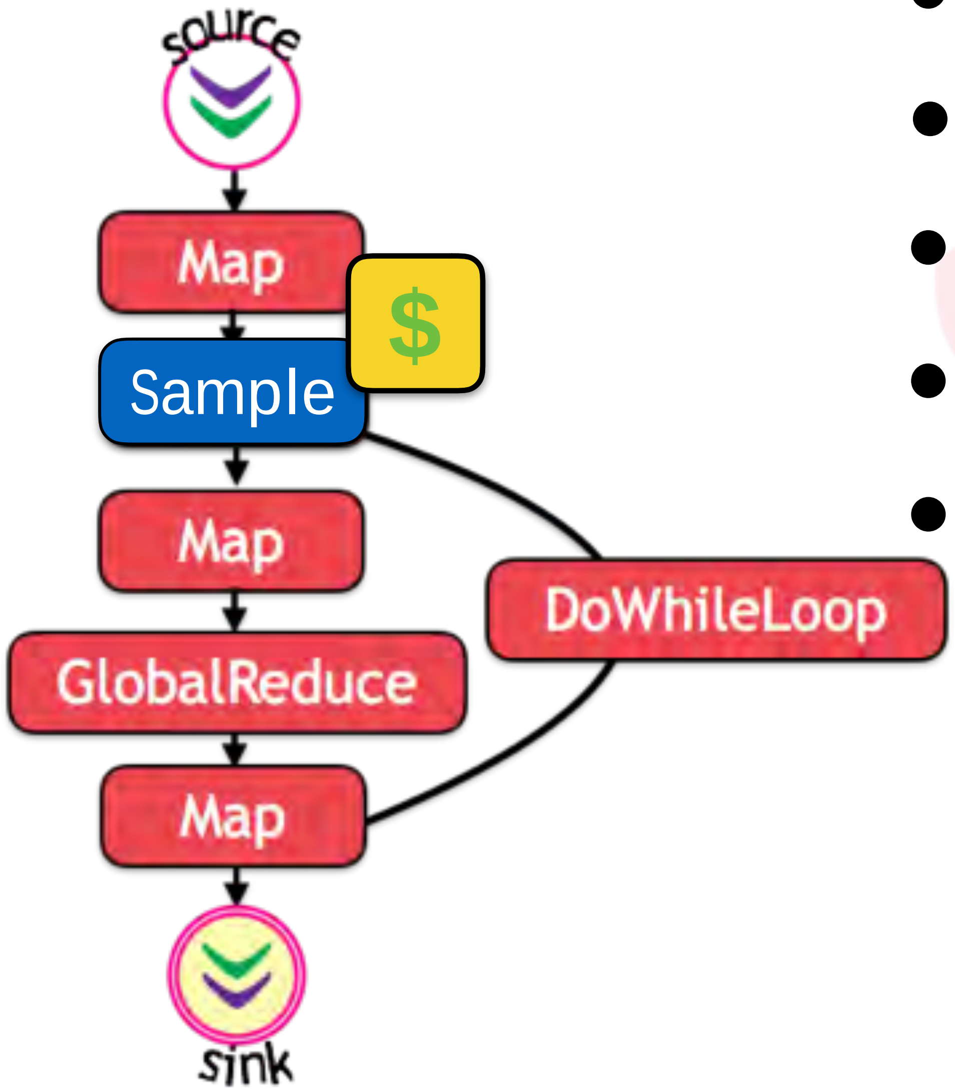
Take Away

Stay Tuned!

http://da.qcri.org/rheem/

https://github.com/daqcri/rheem

- A Cross-Platform System
- Focus on your App and let Rheem do the rest



- Platform-Agnostic Rheem plan
- Platform-Independent Jobs
- Custom Operators
- Cost Functions and Cardinalities
- Adaptive Optimization

JVM Spark

Still To Come…

- Learning Cost Functions
- In-Memory Data Processing
- More Data Processing Platforms
- Cross-Platform Fault-Tolerance