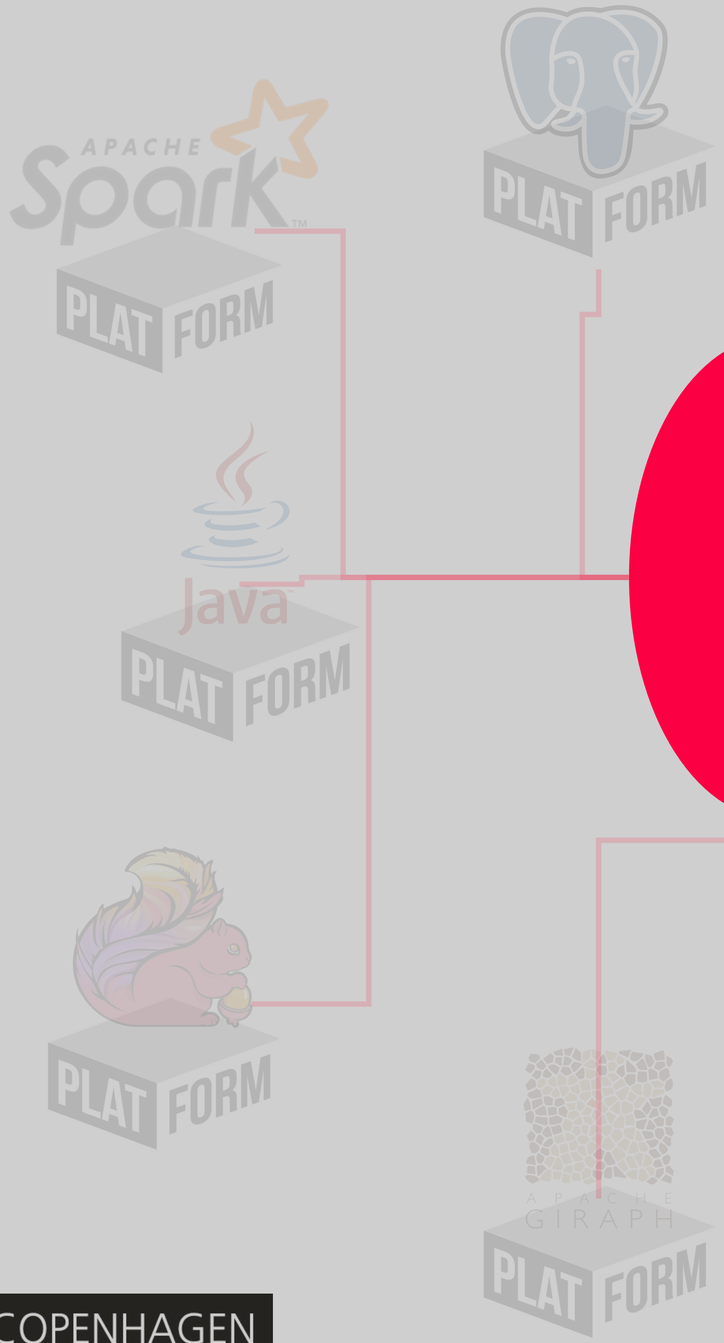


Cross-platform Data Analytics with Apache Wayang

Big Data Management

10-10-2025

In the previous lecture …

- Apache Spark under the hood
  - Jobs, Tasks, Stages
  - Caching
  - Lineage
- Apache Spark APIs
  - Opportunities for optimization

Is Spark always a good choice?

Is Spark always a good choice?

Is Apache Spark always a good choice?

No — **Apache Spark is not always the best choice**.
While it's a powerful and widely used big data processing framework, it's not ideal for every situation.
Let's break this down clearly:

✅ When Spark *is* a good choice

1. **Large-scale data processing**
   - You have **huge datasets (GBs–TBs–PBs)** that don't fit in memory on a single machine.
   - You need distributed computing across many nodes.
2. **Batch + streaming analytics**
   - You want to use one engine for **both historical (batch)** and **real-time (streaming)** data with tools like **Spark Structured Streaming**.
3. **Complex transformations**
   - You're doing **ETL**, **machine learning**, **graph processing**, or **SQL-like analytics** on distributed data.
4. **Integration with the big data ecosystem**
   - You're already using **Hadoop**, **Hive**, **Kafka**, or **Delta Lake**, and Spark fits well into that stack.

Is Spark always a good choice?

❌ **When Spark is *not* a good choice**

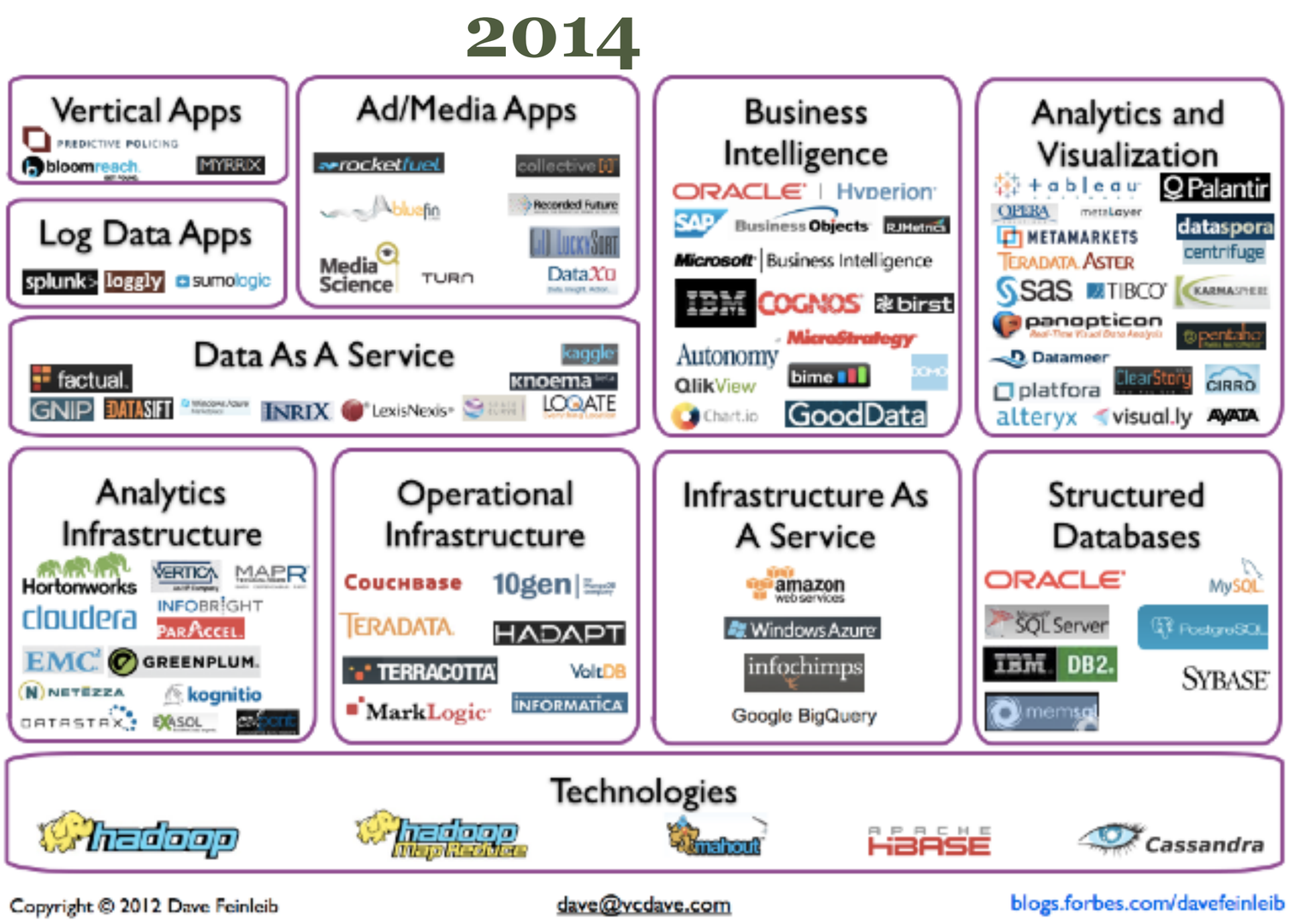
1. **Small or moderate data sizes**
   - If your data **fits comfortably in memory on one machine**, pandas, Dask, or even SQL databases are **simpler and faster**.
2. **Low-latency requirements**
   - Spark is **not real-time** — even Structured Streaming has latency on the order of seconds, not milliseconds.
     → Use **Flink**, **Kafka Streams**, or **Materialize** for sub-second streaming.
3. **Resource constraints**
   - Spark can be **resource-hungry** (CPU, memory, I/O). Running it for small jobs can be **overkill** and **costly**.
4. **Operational complexity**
   - Spark clusters require **setup, tuning, and monitoring**. For simpler workflows, **DuckDB**, **Polars**, or **Snowflake** might be easier.
5. **Interactive or ad-hoc workloads**
   - Spark's startup overhead makes it **slow for quick, interactive analysis** compared to tools like **Jupyter + pandas** or **DuckDB**.

FACT 1

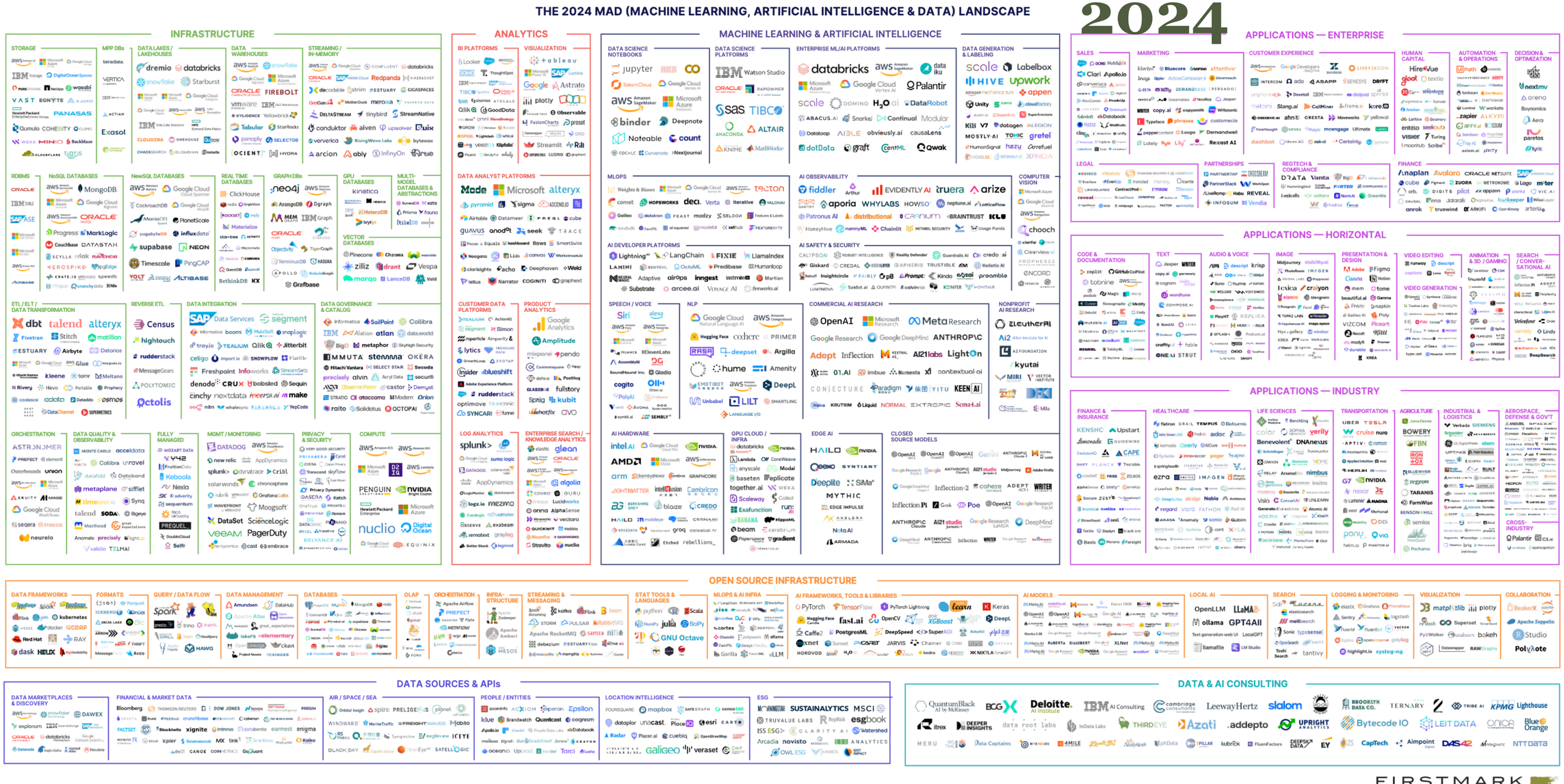
— One Size Does Not Fit All—

Big Data & AI Landscape growth

Big Data & AI Landscape growth



Big Data & AI Landscape growth



Version 1.0 - March 2024 © Matt Turck (@mattturck) , Aman Kabeer (@AmanKabeer11) & FirstMark (@firstmarkcap) Blog post: mattturck.com/MAD2024 Interactive version: MAD.firstmarkcap.com Comments? Email MAD2024@firstmarkcap.com

FIRSTMARK EARLY STAGE VENTURE CAPITAL

FACT 2

– Complexity of Data Analytics–

Data Analytics in 2010's

Relational Data

Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

Key = 24

Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

SQL Queries

```
SELECT name
FROM employee e
INNER JOIN Person p
ON p.firstname = e.name
```

Data Analytics in 2020's

Relational Data

Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

Key = 24

Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

Text Data

Streaming Data

Graph data

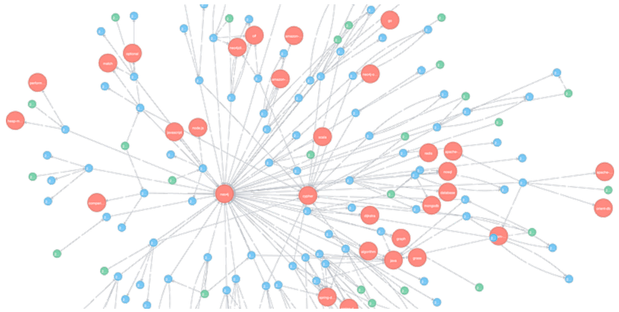
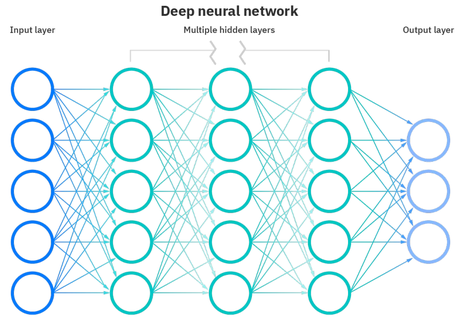


Image data

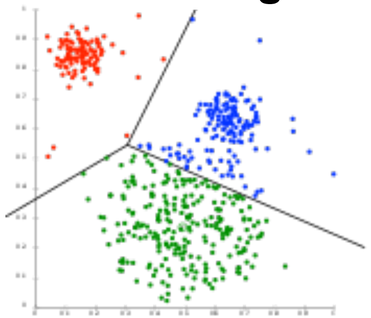
SQL Queries

```
SELECT name
FROM employee e
INNER JOIN Person p
ON p.firstname = e.name
```

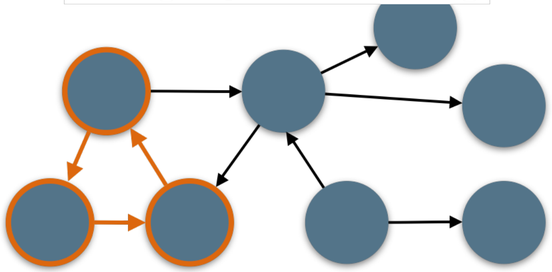
Deep Learning



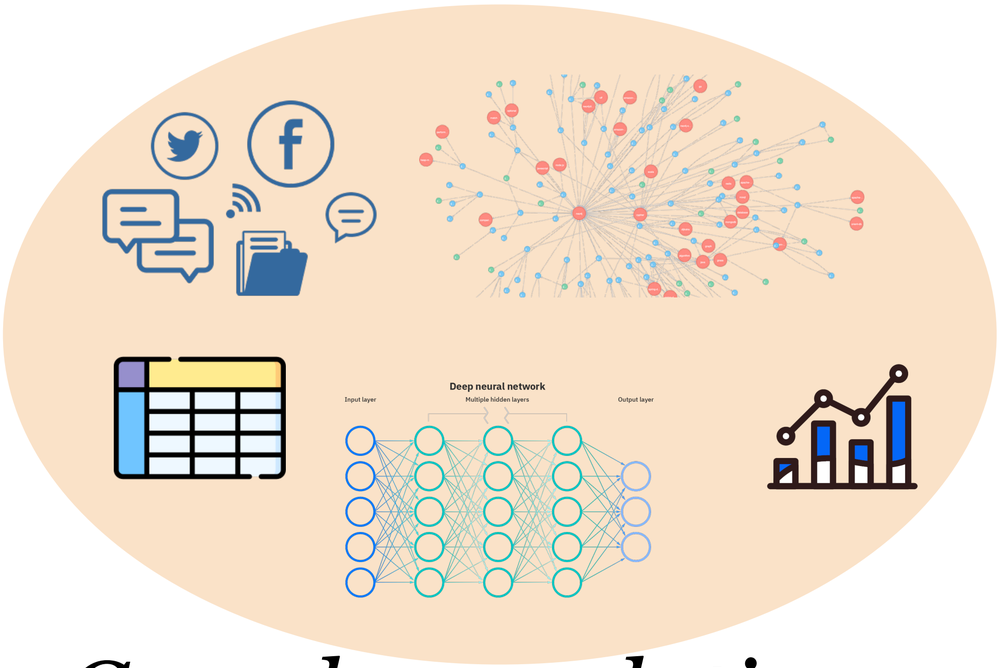
Clustering



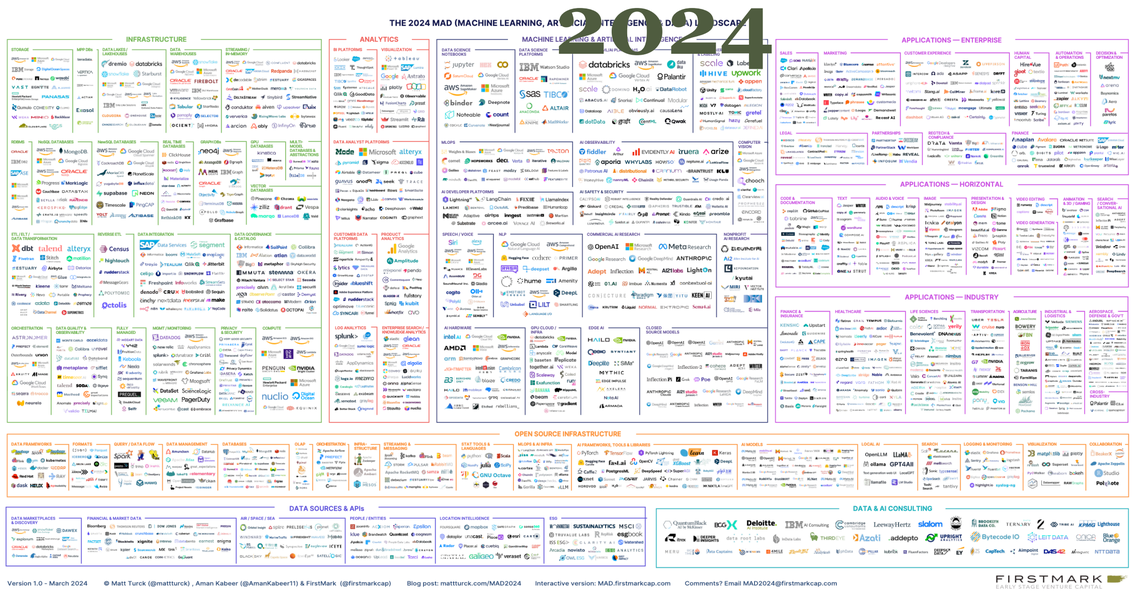
Subgraph Matching



Users/Developers Overwhelmed



Complex analytics



Zoo of systems

Current Approaches

Custom Systems & ETL/ELT processes

Current Approaches

Custom Systems & ETL/ELT processes

Hybrid Systems

HadoopDB, SystemDS, FlumeJava

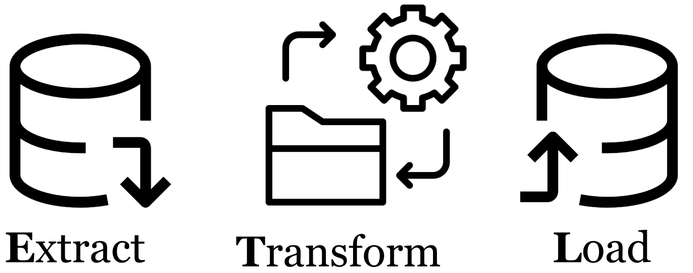
Current Approaches

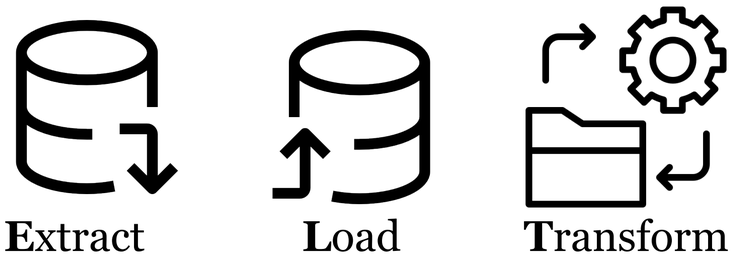
Custom Systems & ETL/ELT processes

Hybrid Systems

HadoopDB, SystemDS, FlumeJava

Complex Data Architectures and Infrastructures





Current Approaches

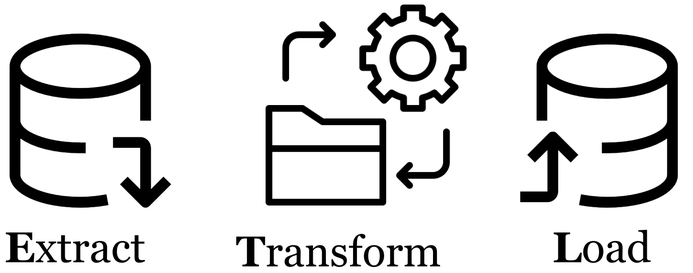
Custom Systems & ETL/ELT processes

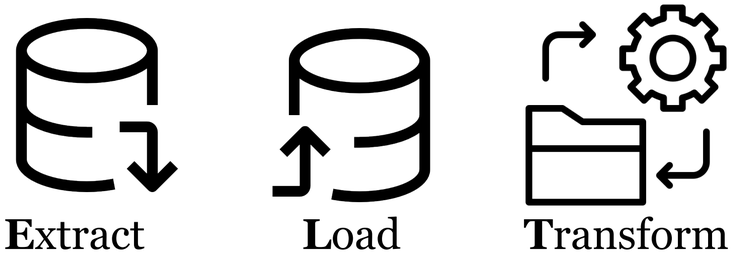
Hybrid Systems

HadoopDB, SystemDS, FlumeJava

Not Maintainable

Complex Data Architectures and Infrastructures





Current Approaches

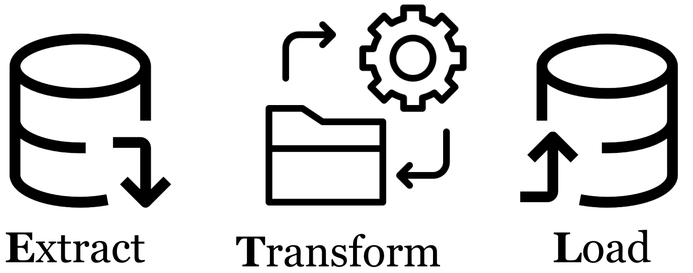
Custom Systems & ETL/ELT processes

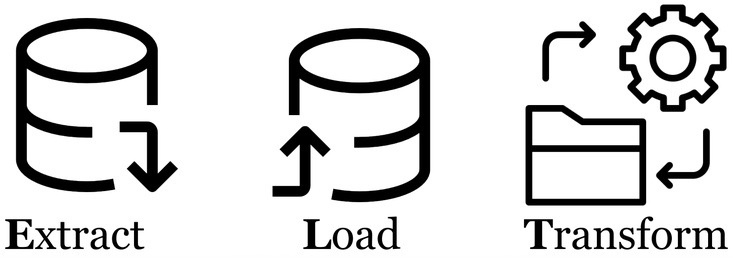
Hybrid Systems

HadoopDB, SystemDS, FlumeJava

Not Maintainable

Complex Data Architectures and Infrastructures





Not Optimal

Costly

Current Approaches

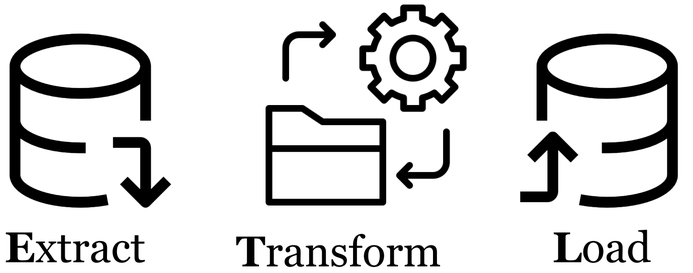
Custom Systems & ETL/ELT processes

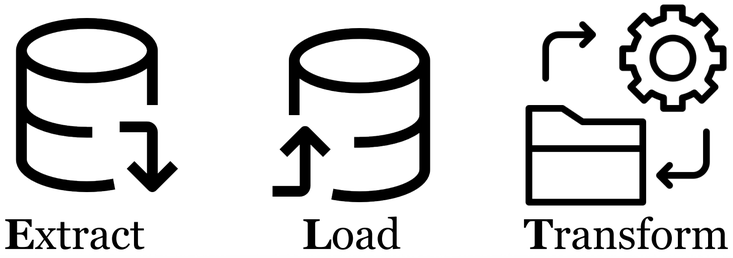
Hybrid Systems

HadoopDB, SystemDS, FlumeJava

Not Maintainable

Complex Data Architectures and Infrastructures





Not Optimal

Costly

Current Approaches

Custom Systems & ETL/ELT processes

*Corporate developers spend approximately **65%** of their effort building bridges between applications (Gartner)*

Not Maintainable

Not Optimal

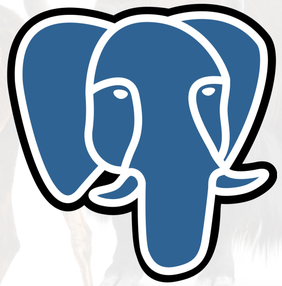
Costly

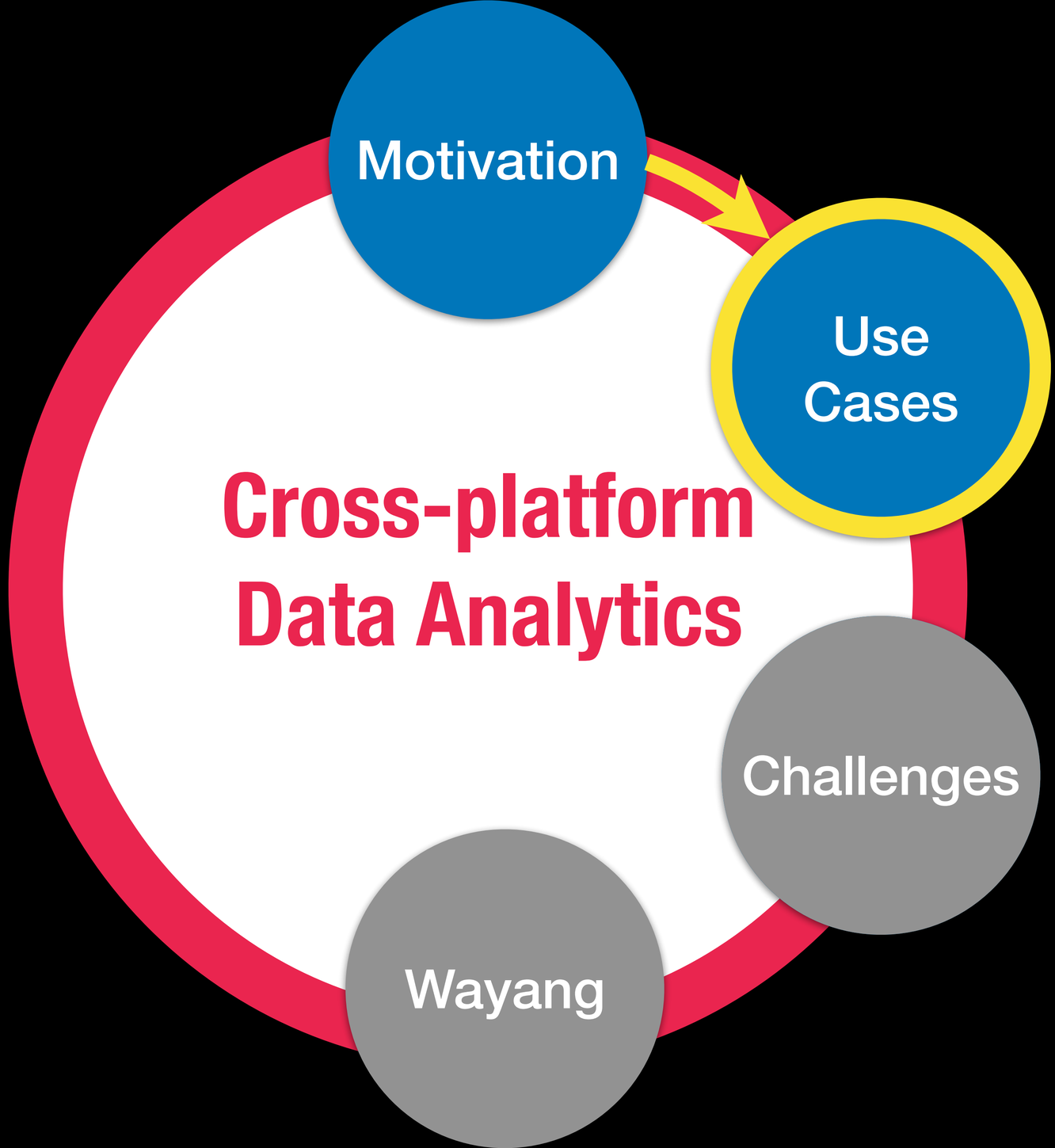
Goal: Unify Data Analytics in One Framework

Goal: Unify Data Analytics in One Framework

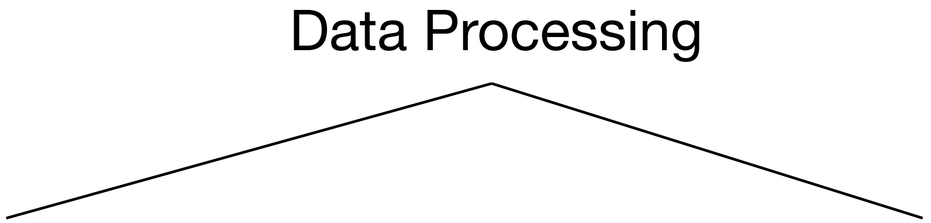
use of multiple data processing platforms (engines) for data analytics

Cross-platform Data Analytics

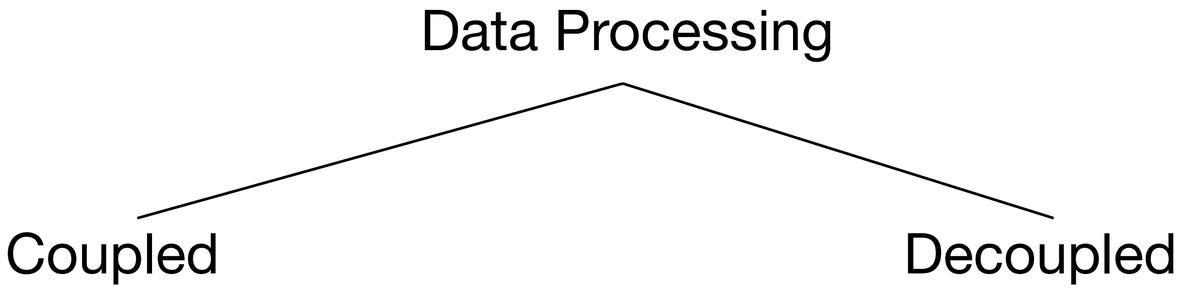



Motivation
Use Cases
Cross-platform Data Analytics
Challenges
Wayang

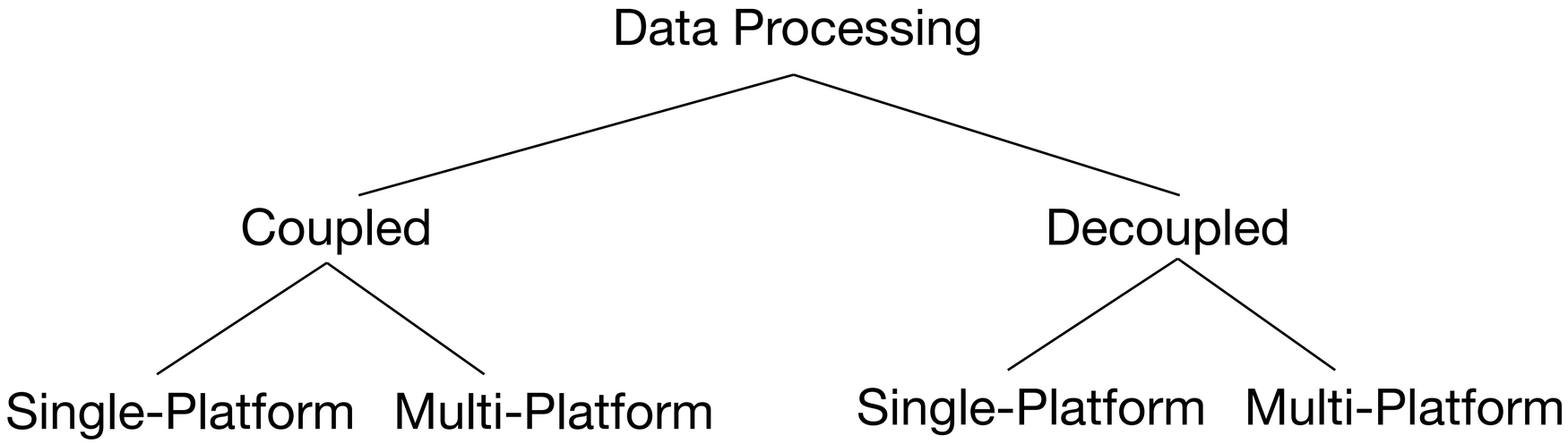
Data Processing Taxonomy



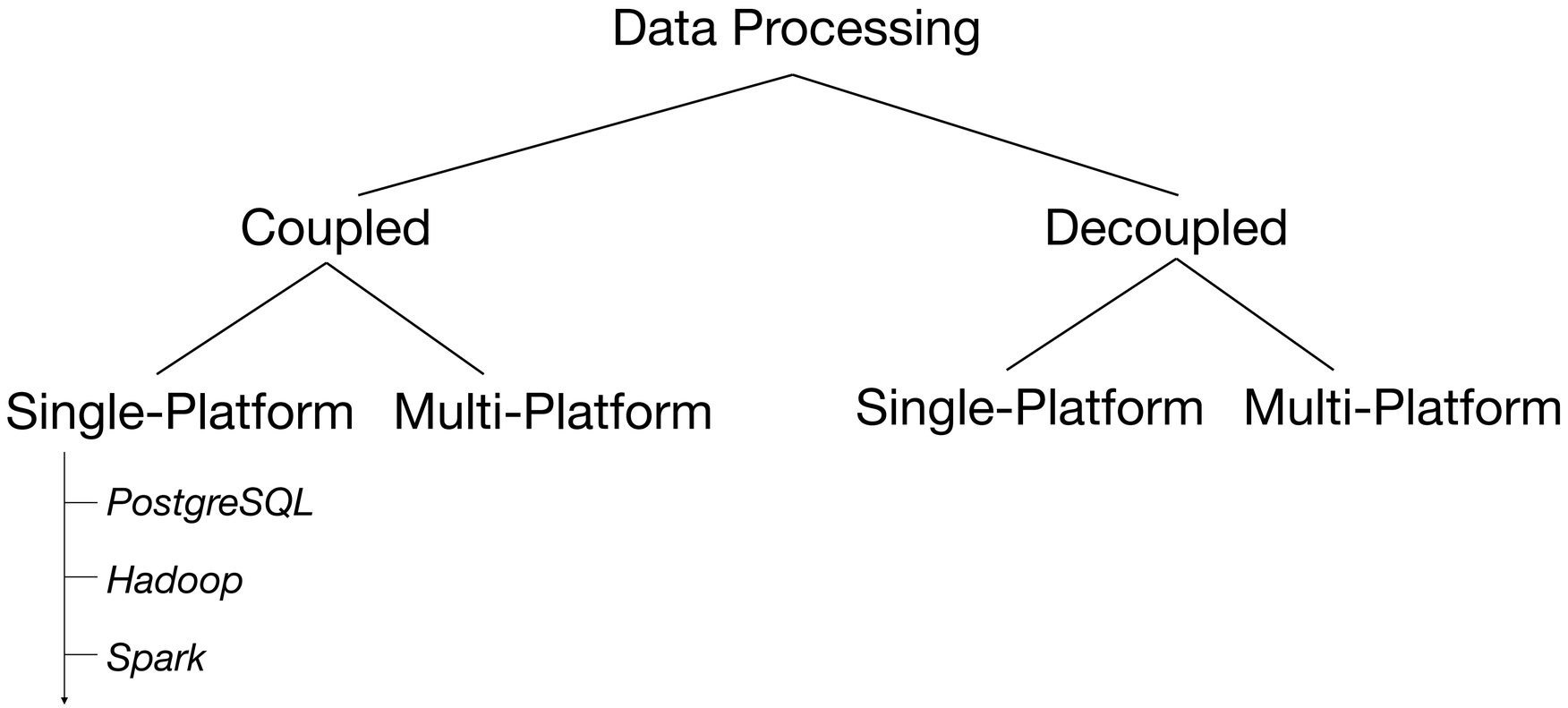
Data Processing Taxonomy



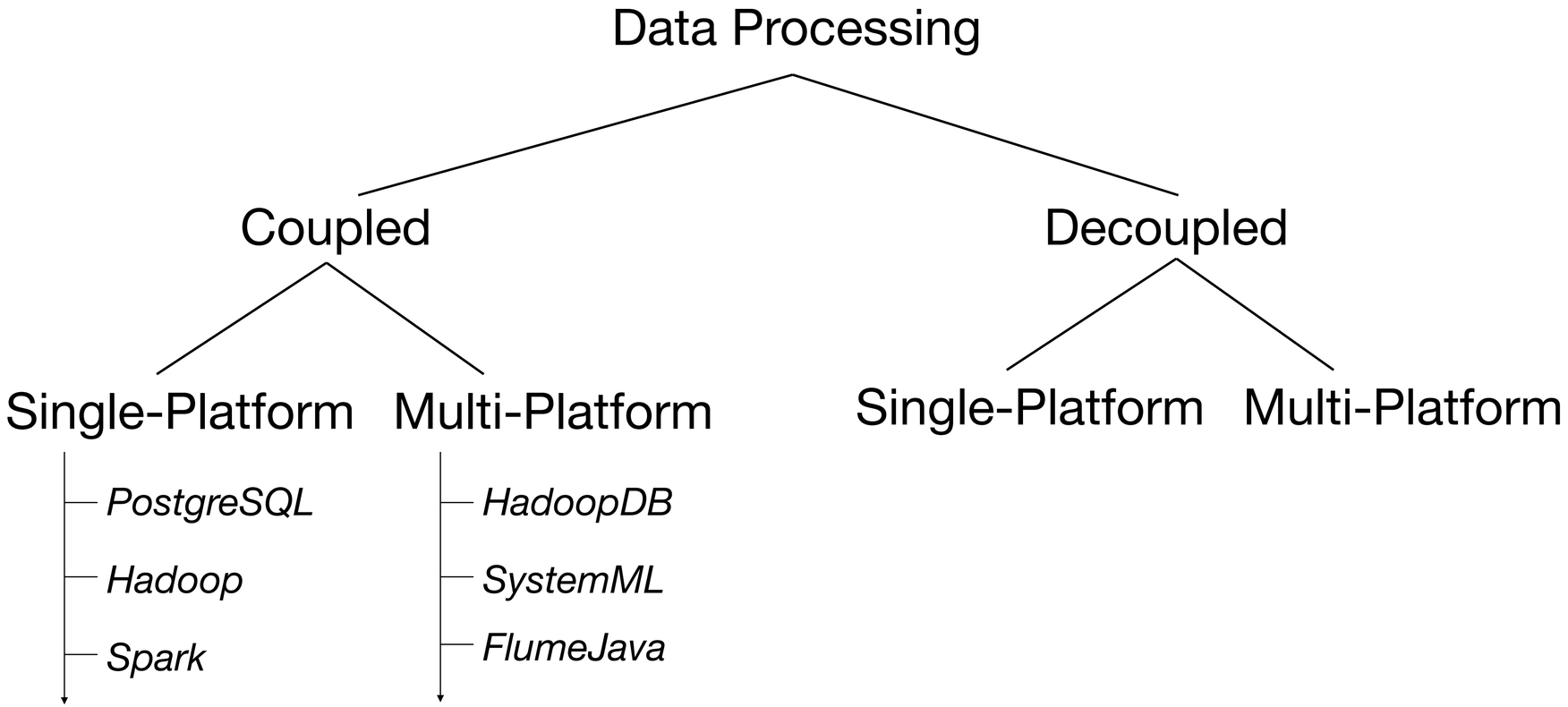
Data Processing Taxonomy



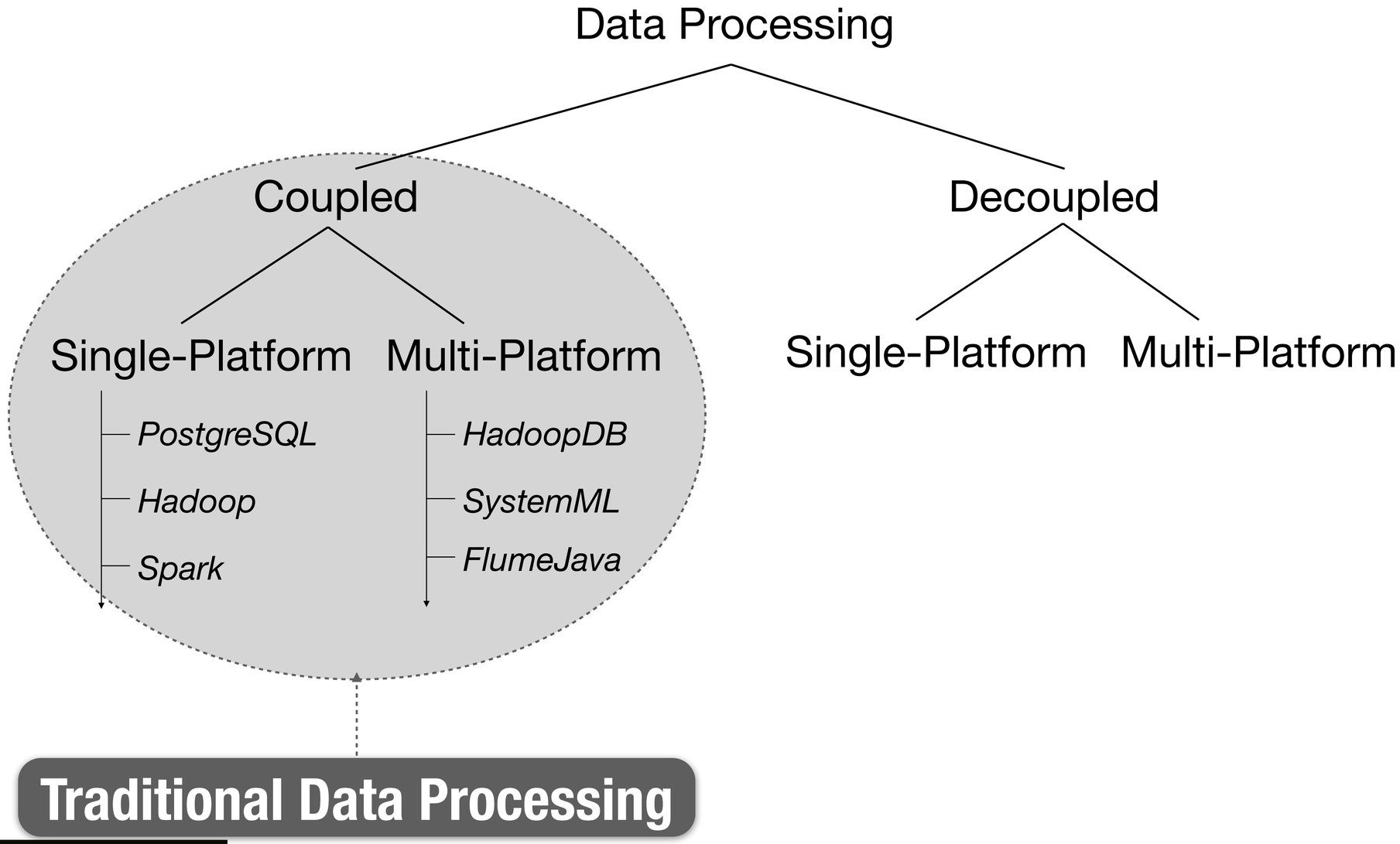
Data Processing Taxonomy



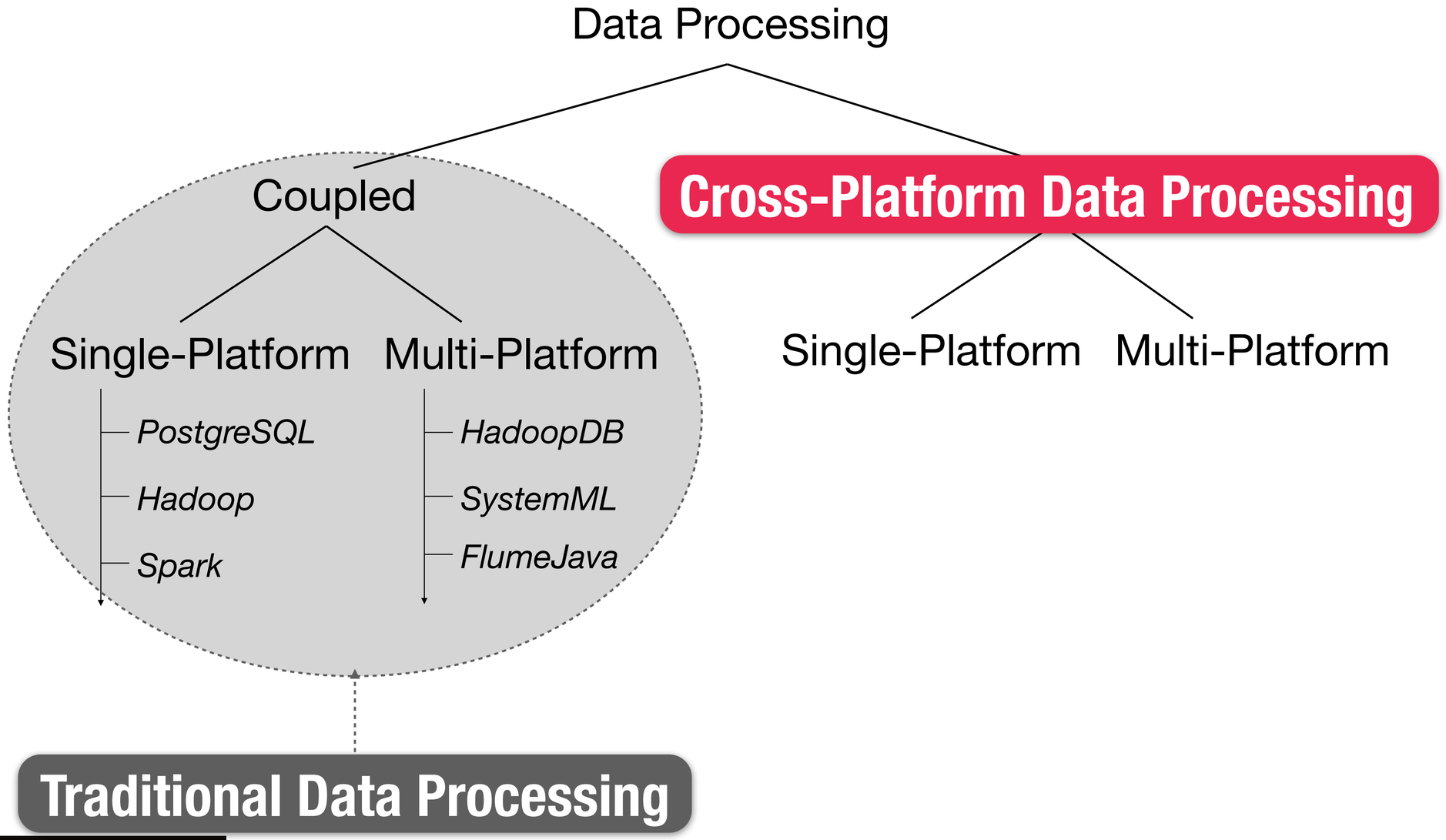
Data Processing Taxonomy



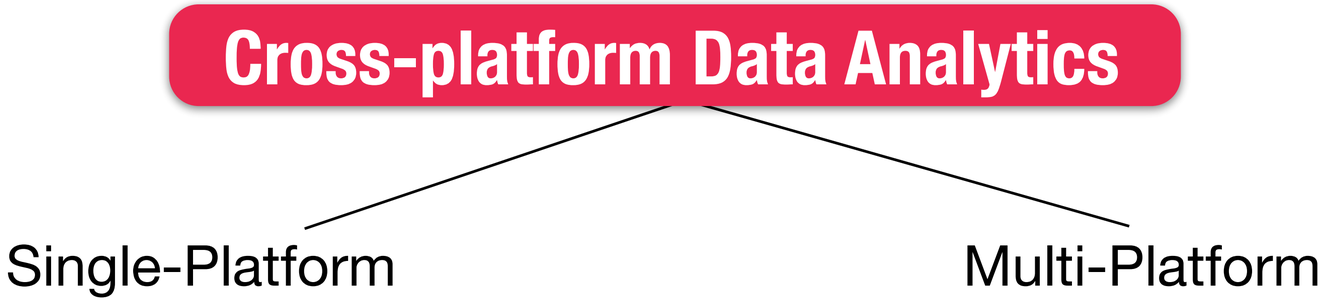
Data Processing Taxonomy



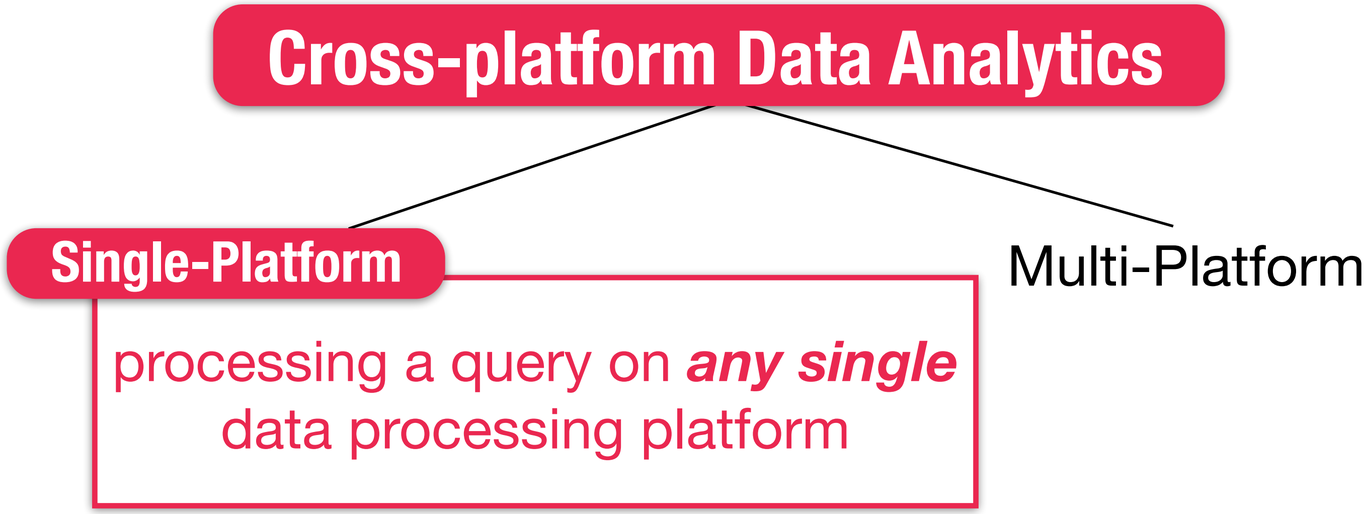
Data Processing Taxonomy



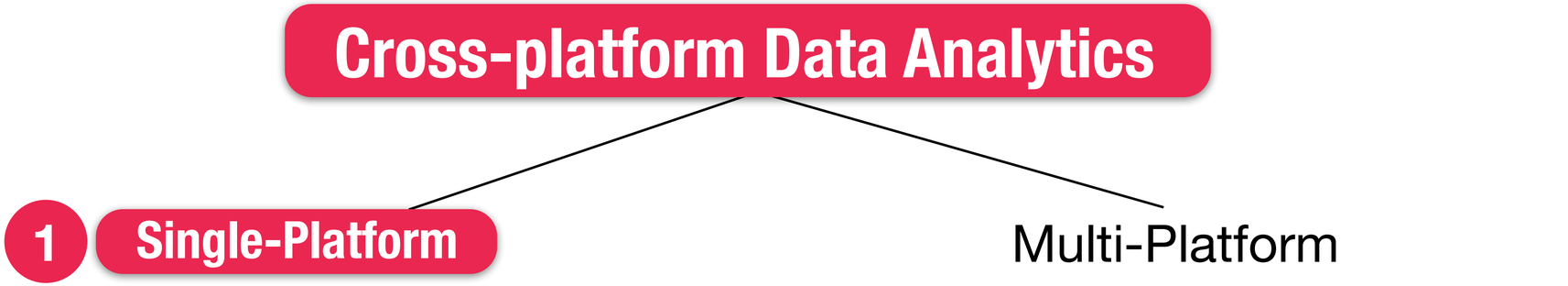
Cross-platform Data Processing Taxonomy



Cross-platform Data Processing Taxonomy

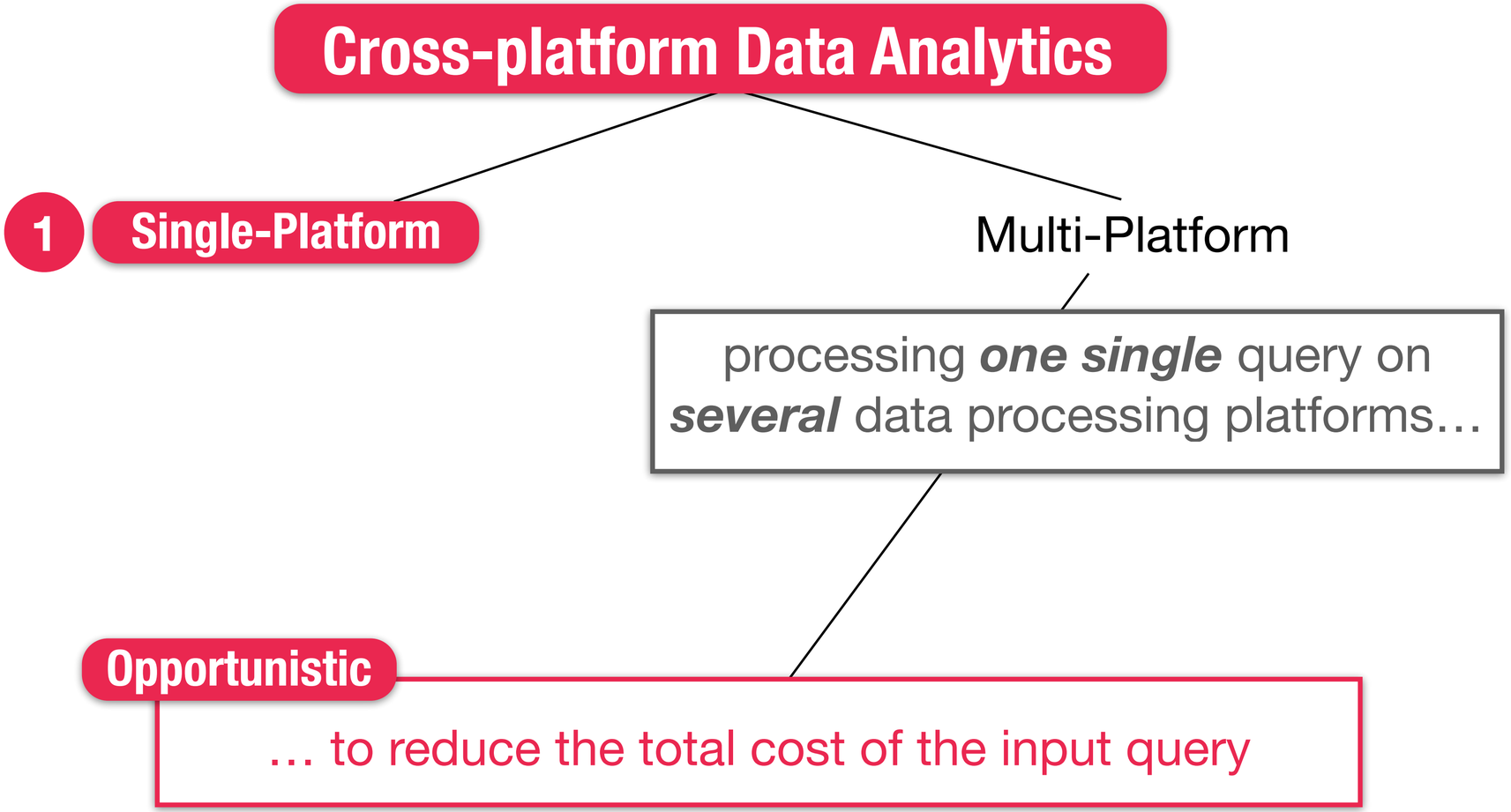


Cross-platform Data Processing Taxonomy

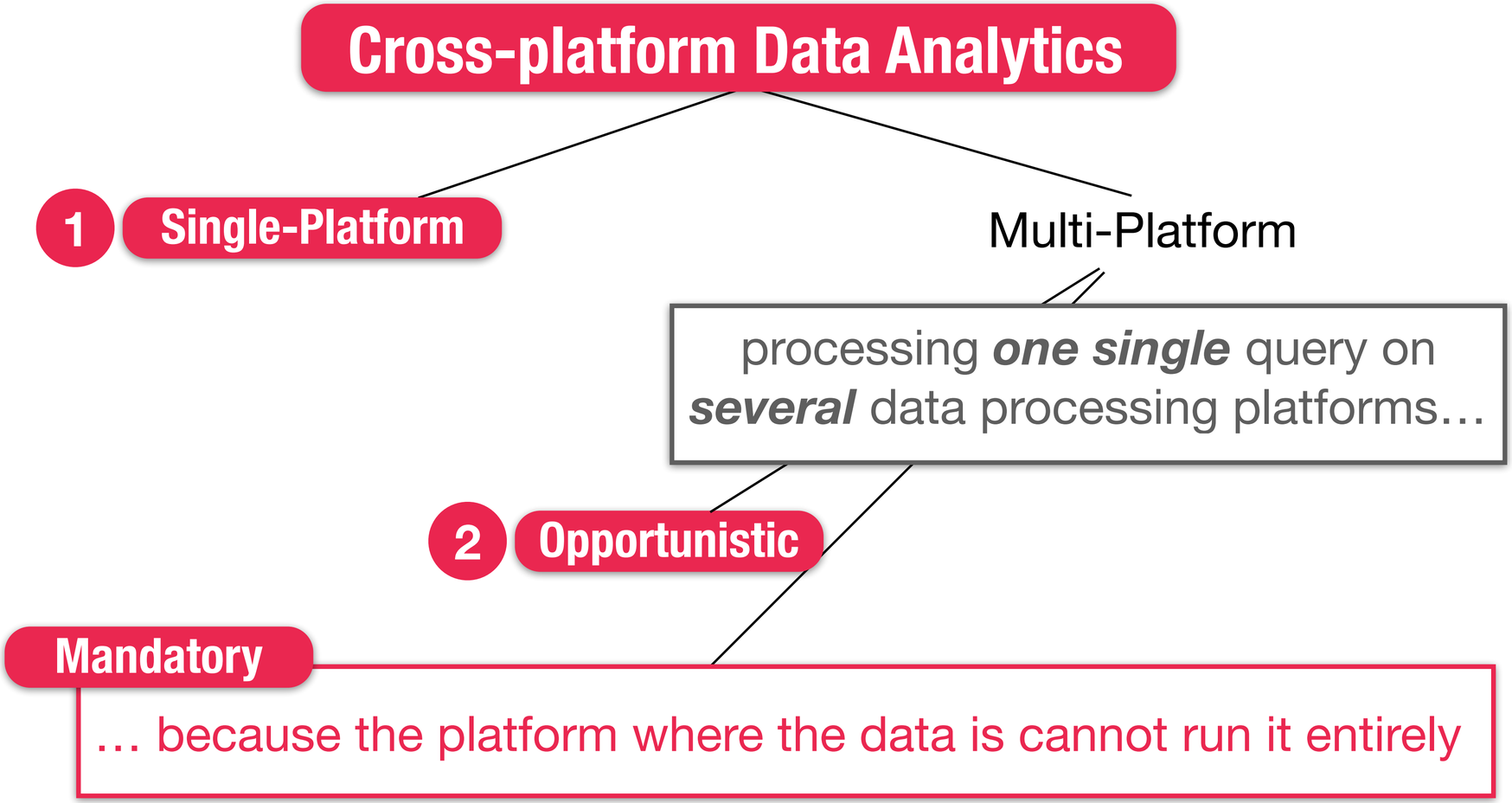


processing ***one single*** query on ***several*** data processing platforms…

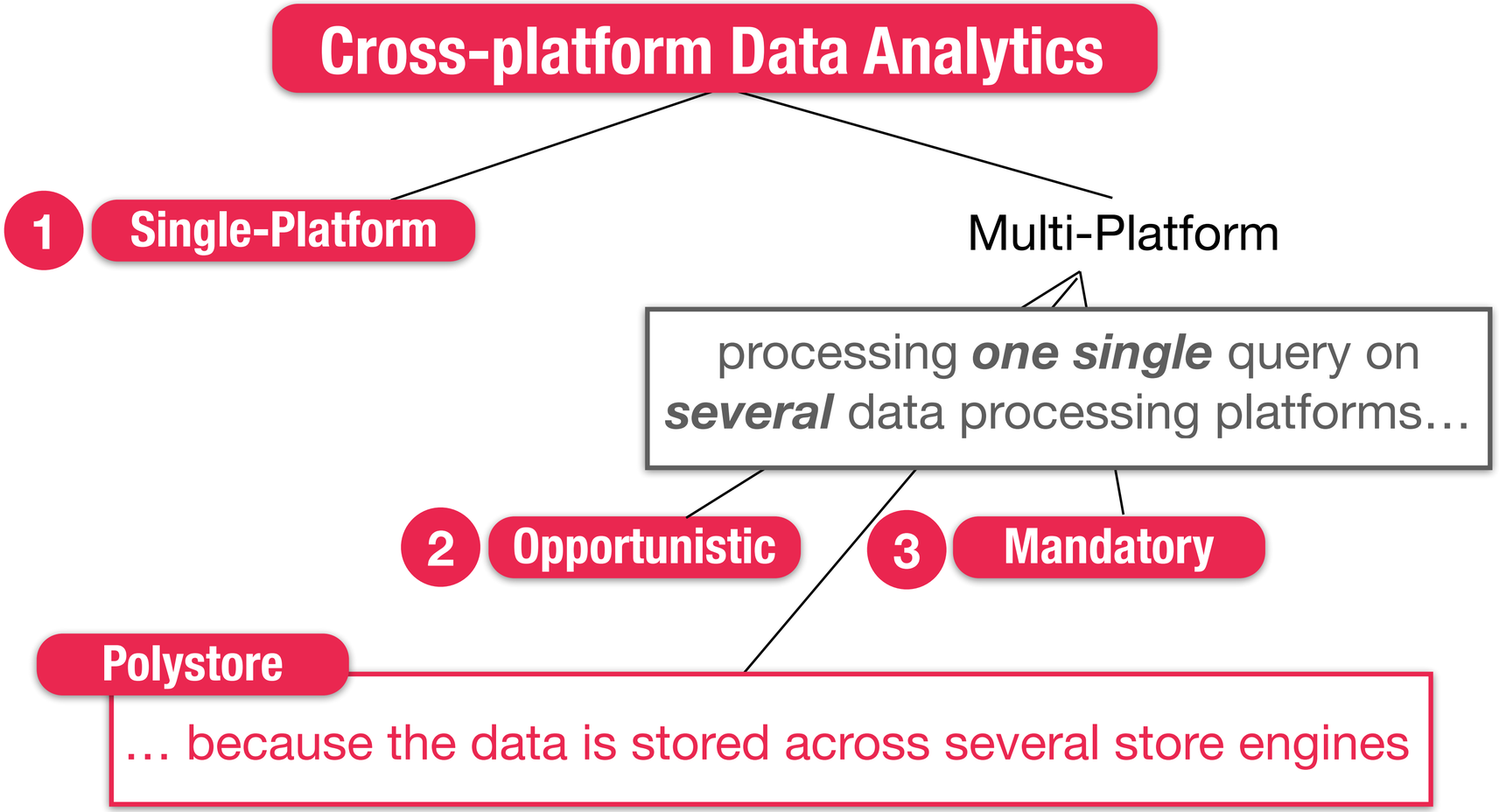
Cross-platform Data Processing Taxonomy



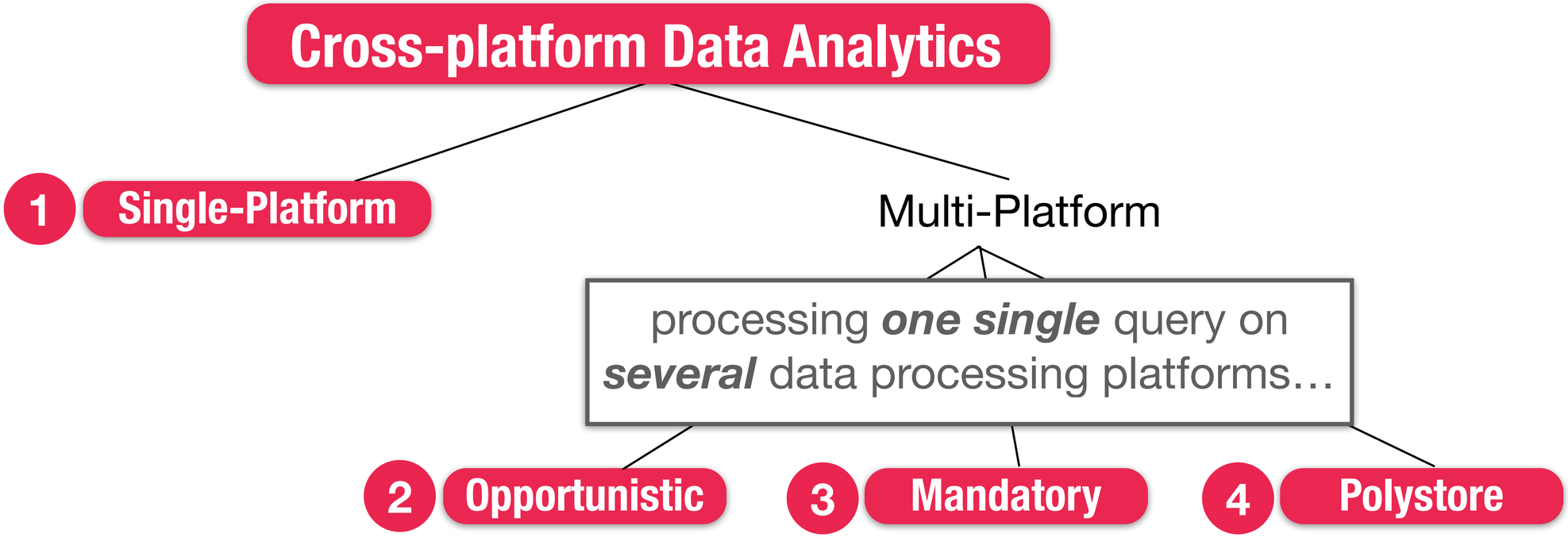
Cross-platform Data Processing Taxonomy



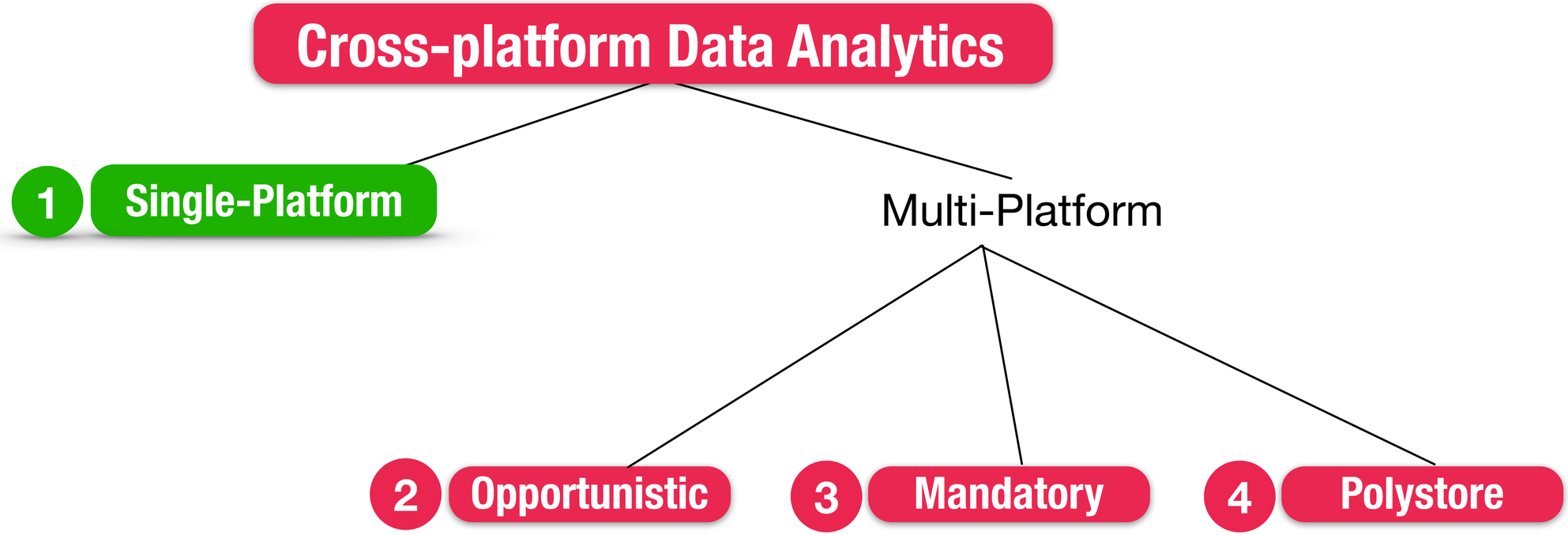
Cross-platform Data Processing Taxonomy



Cross-platform Data Processing Taxonomy



Cross-platform Data Processing Taxonomy



Cross-platform Data Analytics — Single-Platform

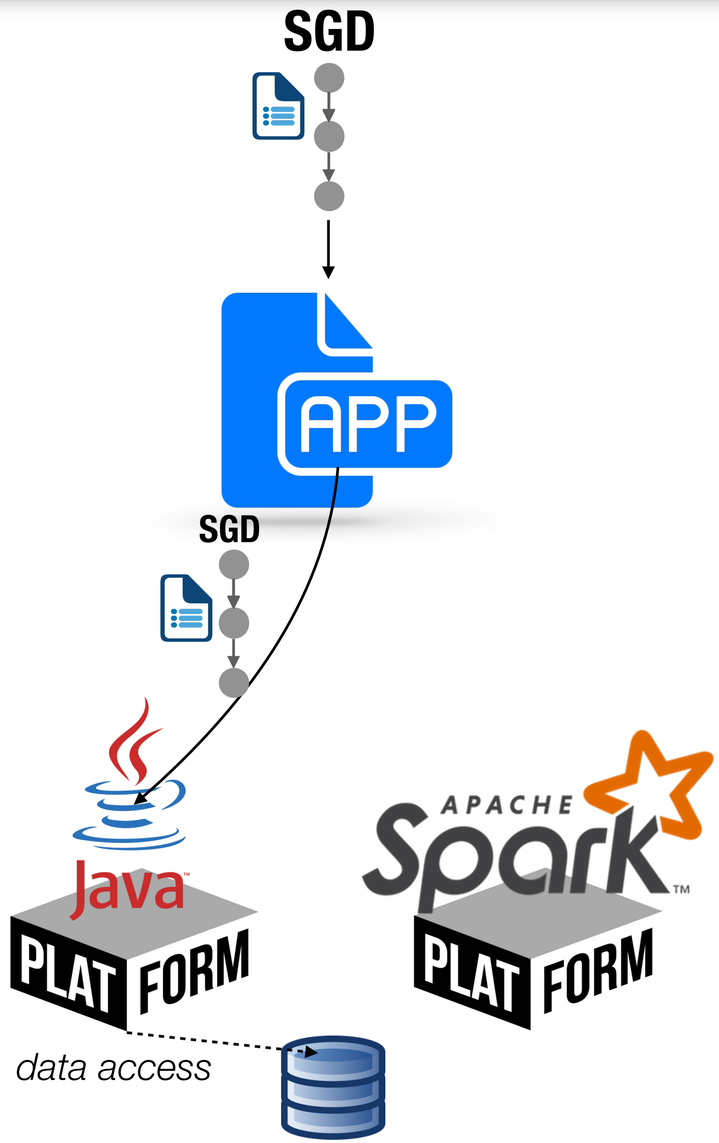
using ***any single*** data processing platform to process a query

Cross-platform Data Analytics — Single-Platform

using ***any single*** data processing platform to process a query

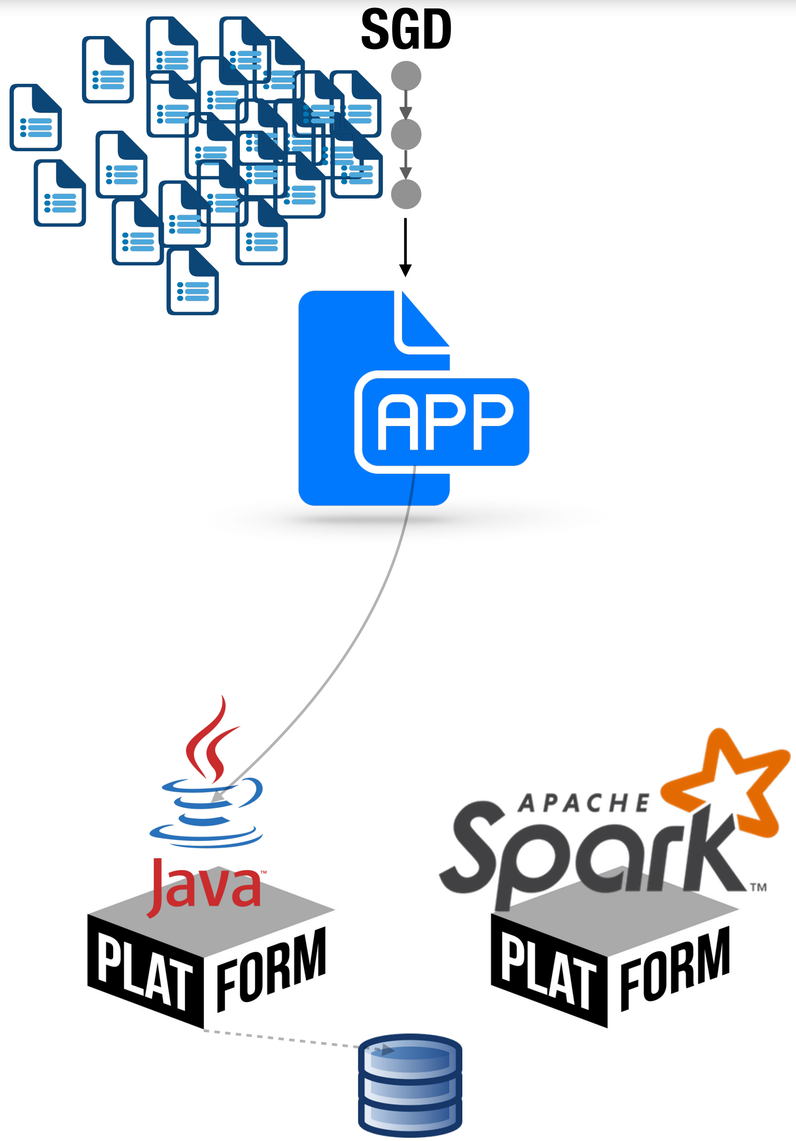
Cross-platform Data Analytics — Single-Platform

using ***any single*** data processing platform to process a query



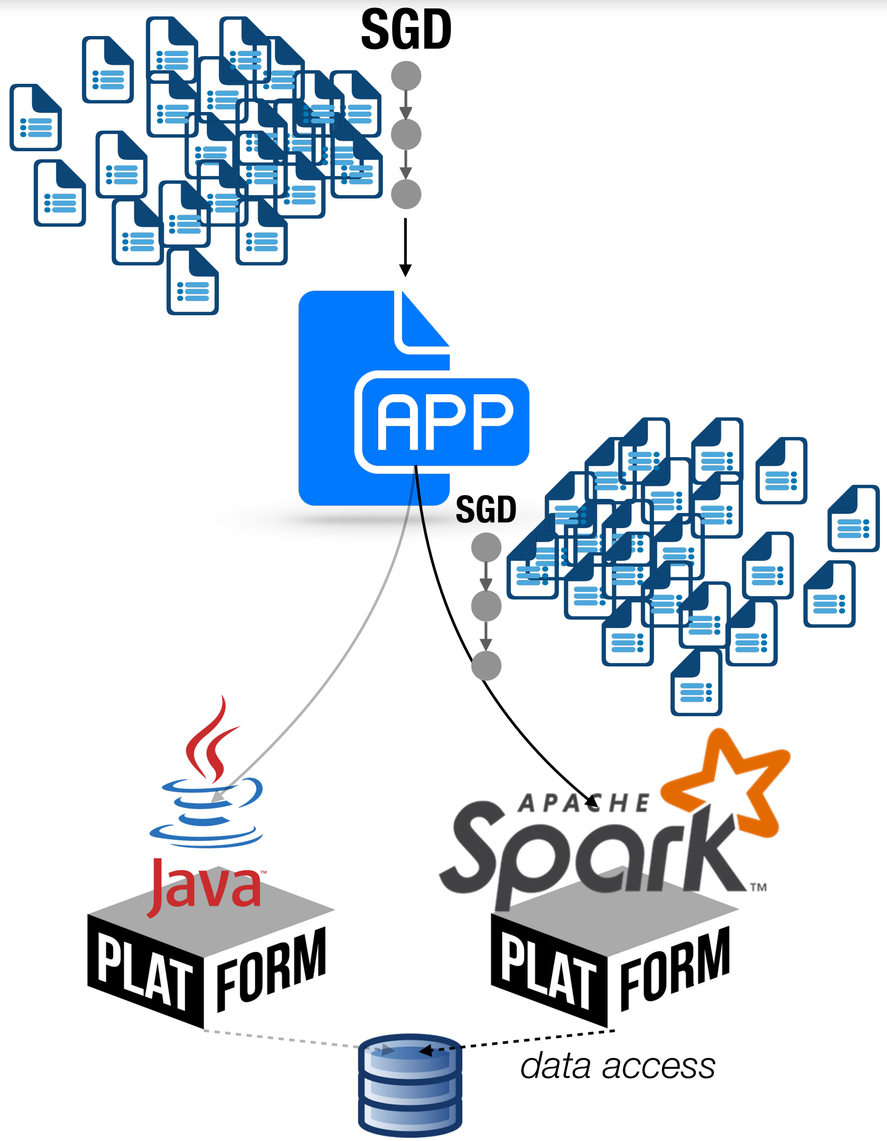
Cross-platform Data Analytics — Single-Platform

using ***any single*** data processing platform to process a query



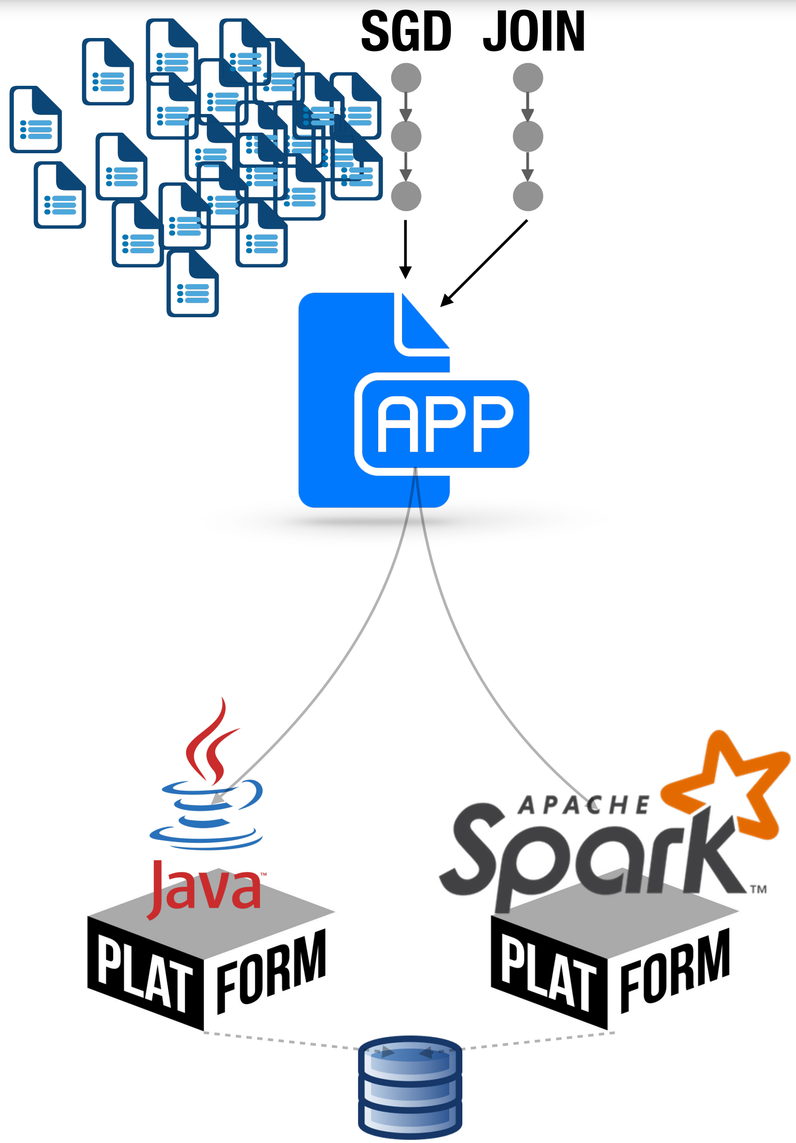
Cross-platform Data Analytics — Single-Platform

using ***any single*** data processing platform to process a query



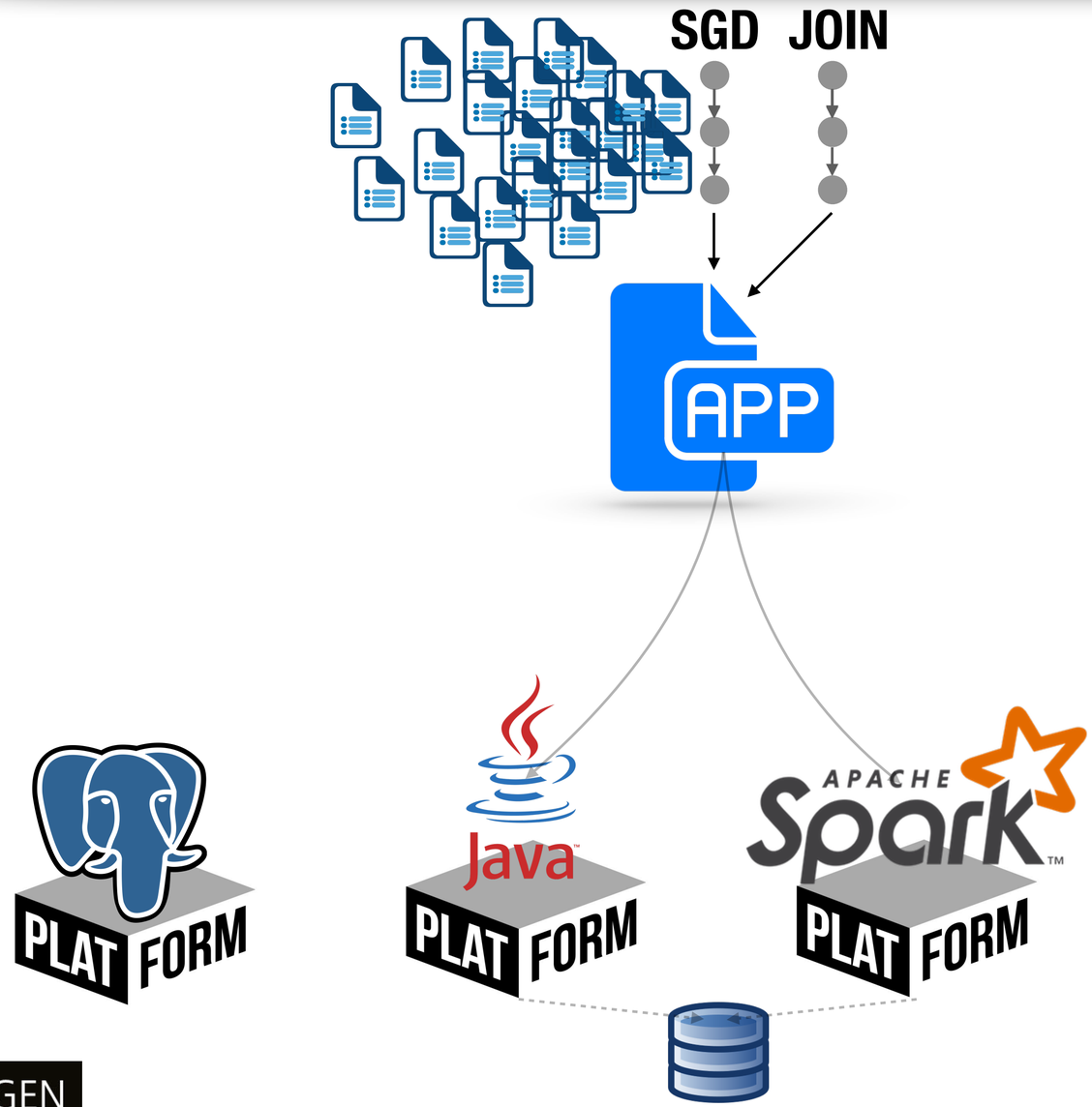
Cross-platform Data Analytics — Single-Platform

using ***any single*** data processing platform to process a query



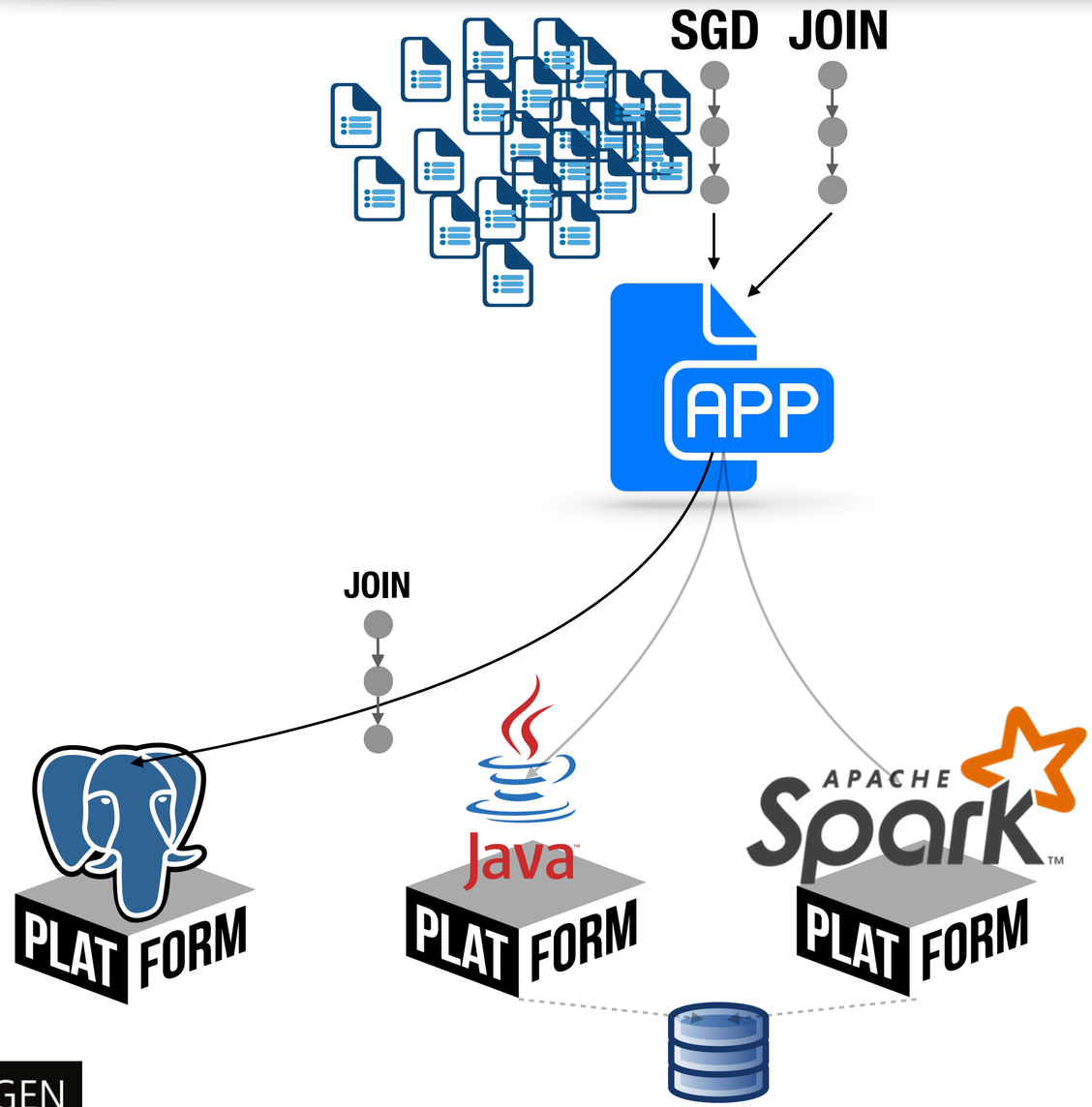
Cross-platform Data Analytics — Single-Platform

using ***any single*** data processing platform to process a query



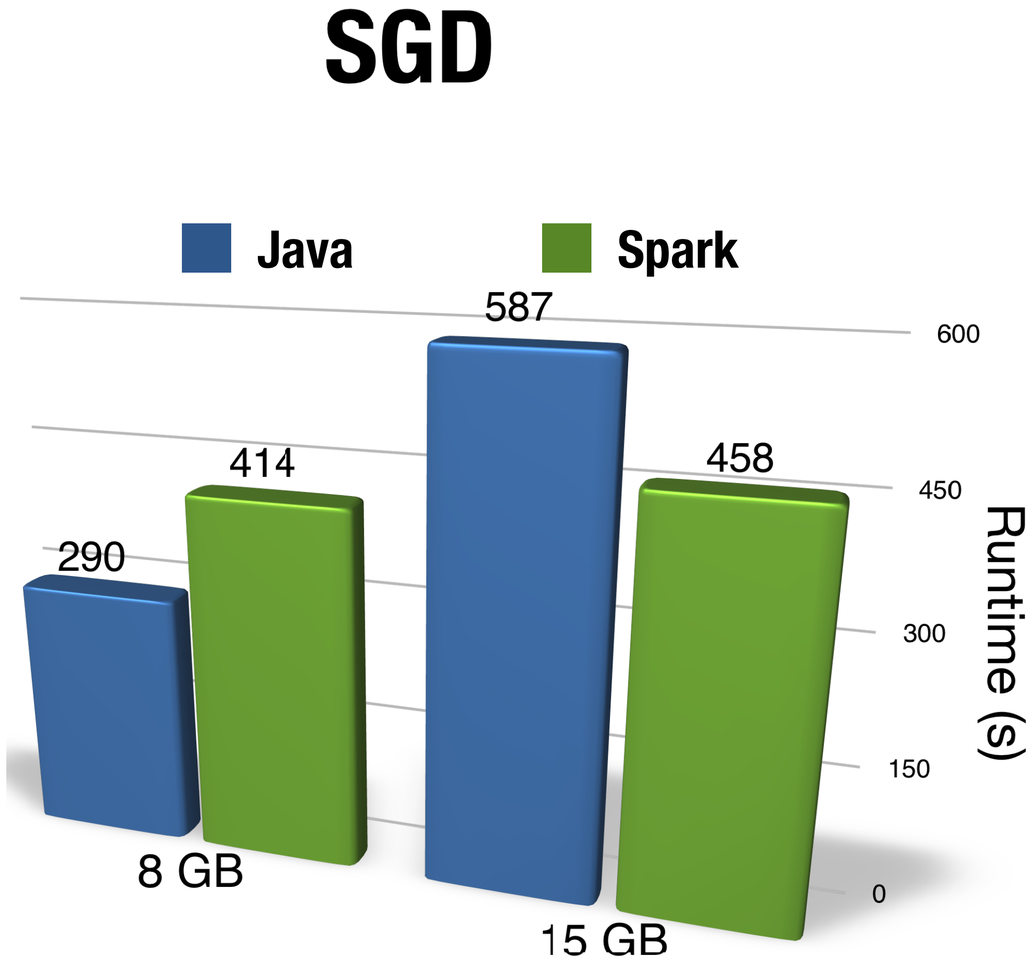
Cross-platform Data Analytics — Single-Platform

using ***any single*** data processing platform to process a query



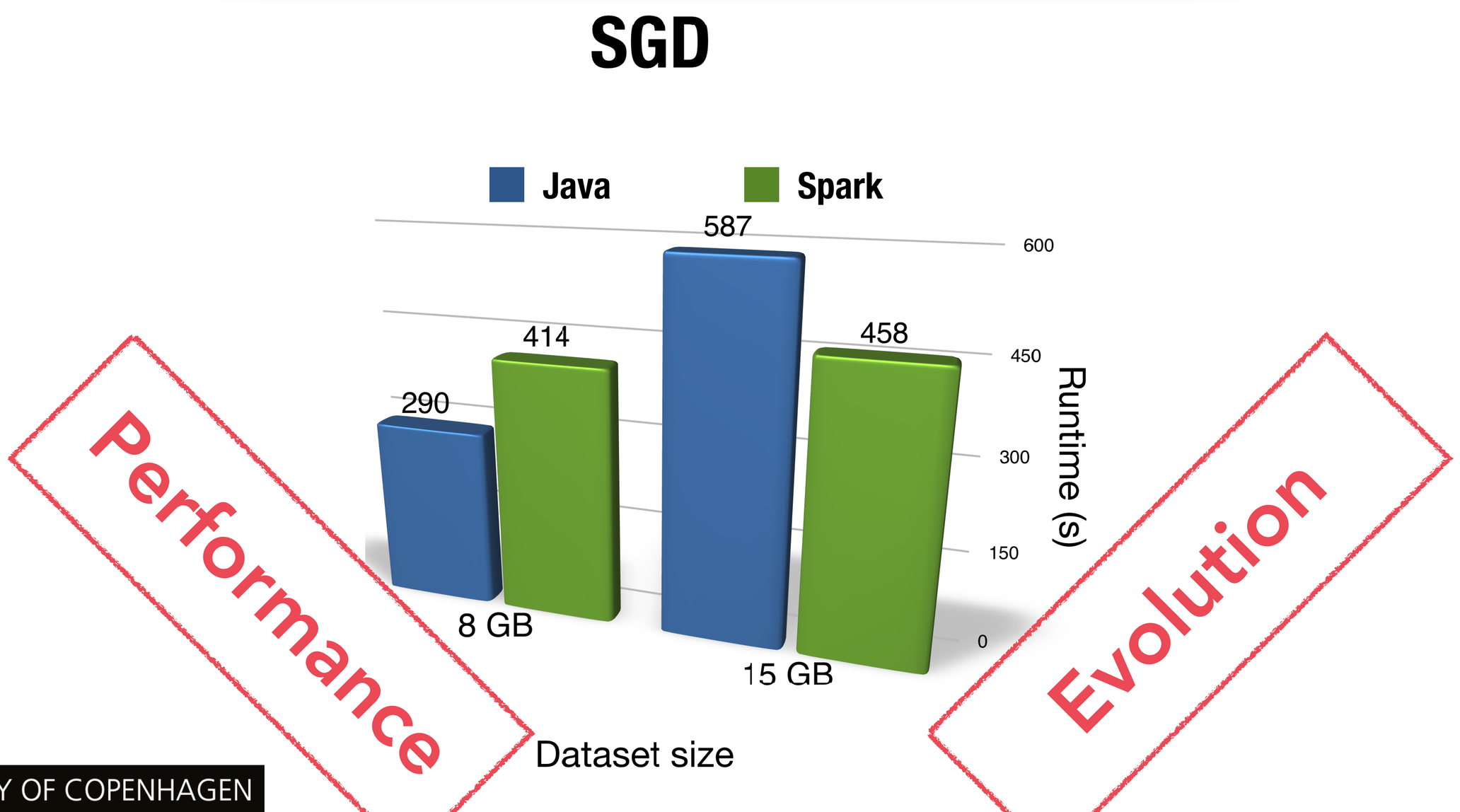
Cross-platform Data Analytics — Single-Platform

using ***any single*** data processing platform to process a query



Cross-platform Data Analytics — Single-Platform

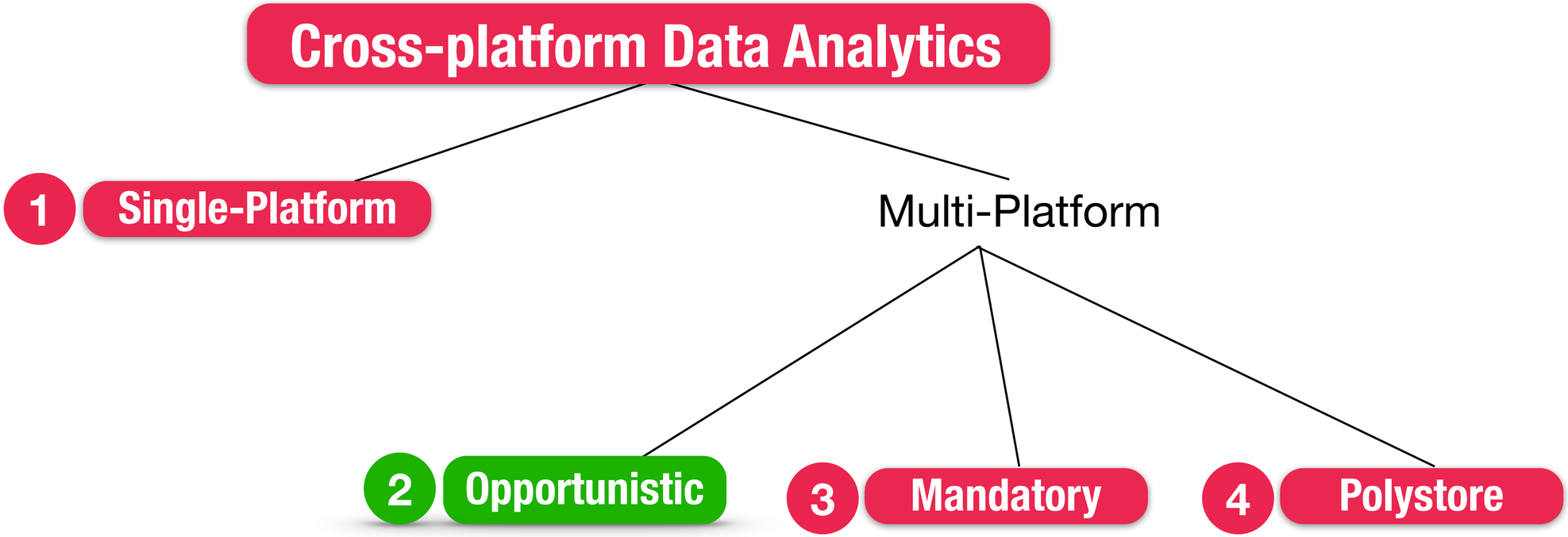
using ***any single*** data processing platform to process a query



System-Store Quadrant

	Monoplatform	Polyplatform
Monostore	Single-Platform	
Polystore	Single-Platform	

Cross-platform Data Processing Taxonomy

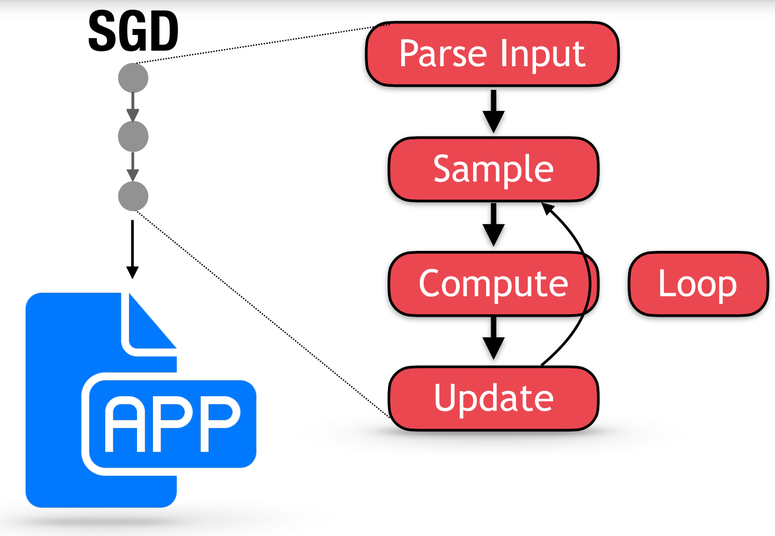


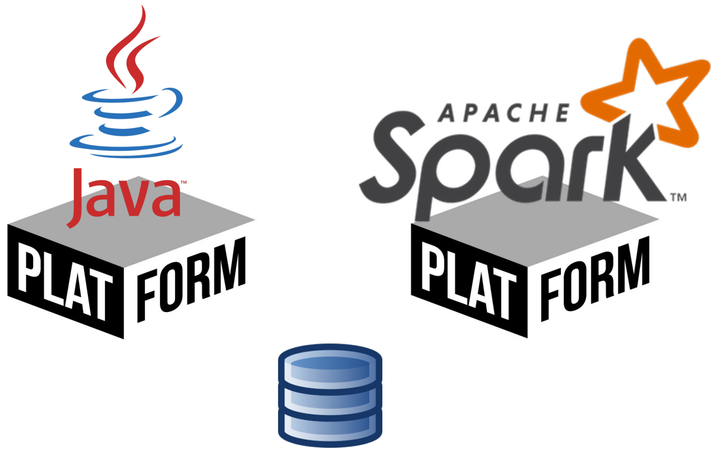
Cross-platform Data Analytics — Opportunistic

using ***several*** data processing platforms to reduce the cost of a query

Cross-platform Data Analytics — Opportunistic

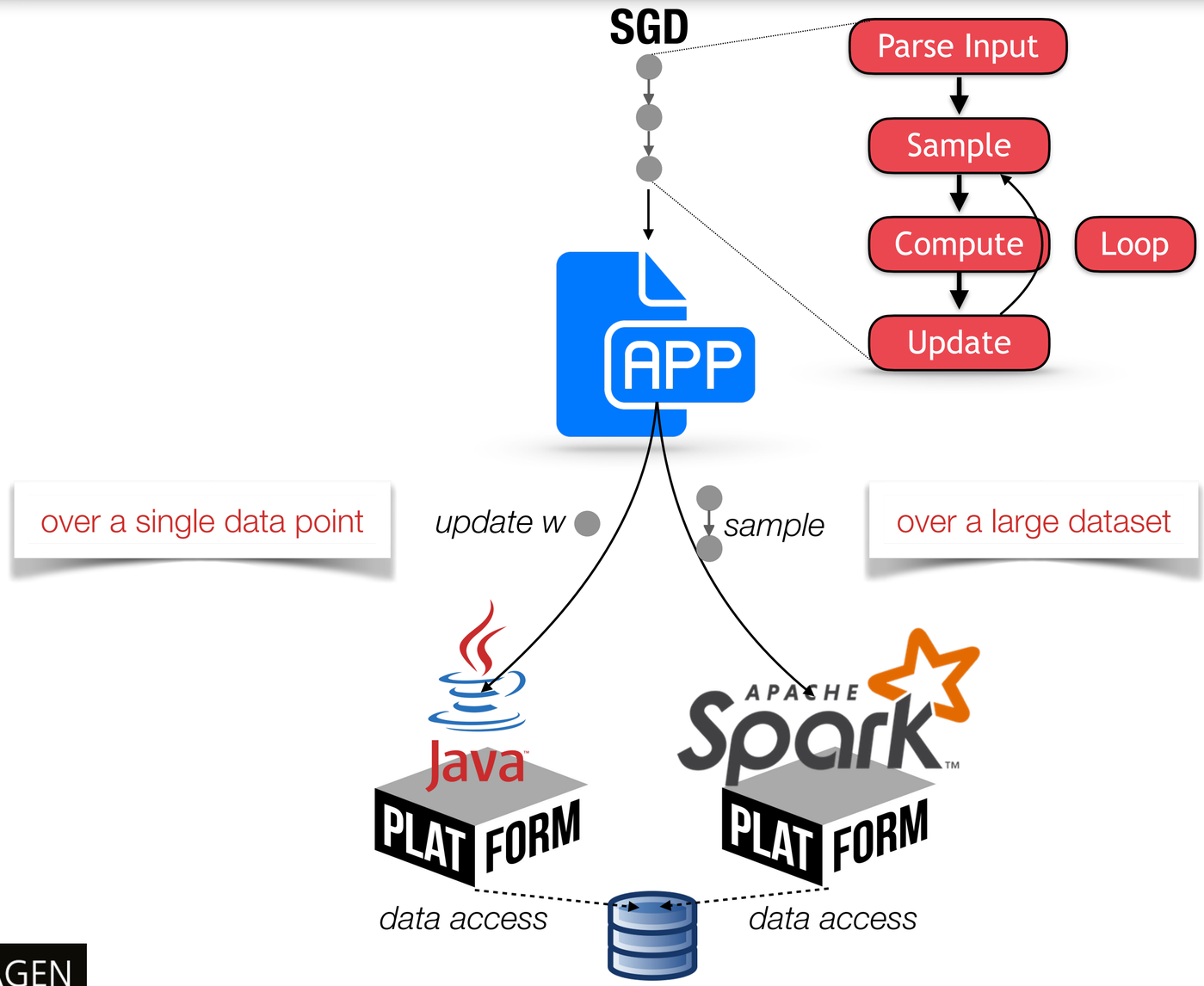
using ***several*** data processing platforms to reduce the cost of a query





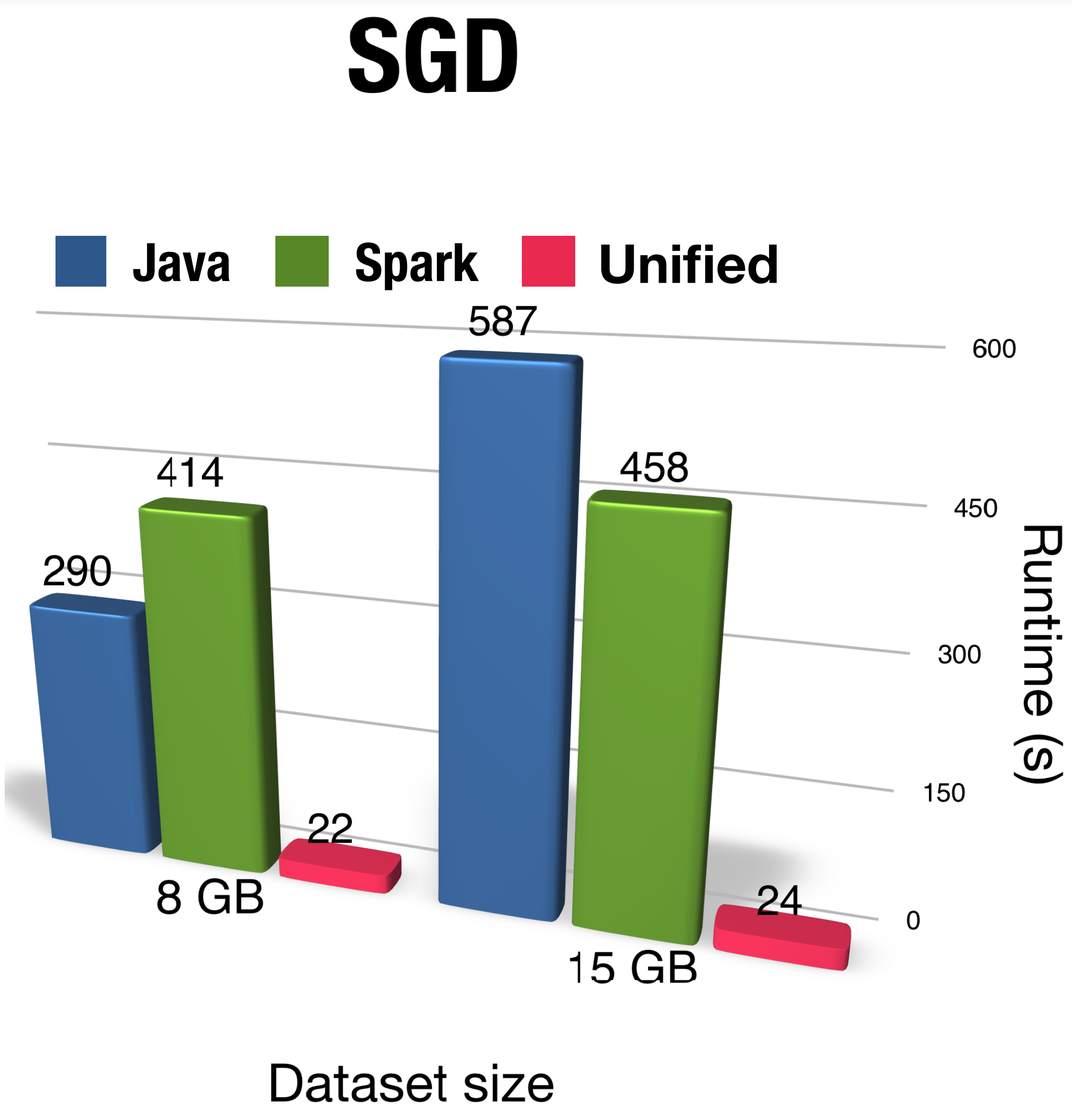
Cross-platform Data Analytics — Opportunistic

using ***several*** data processing platforms to reduce the cost of a query



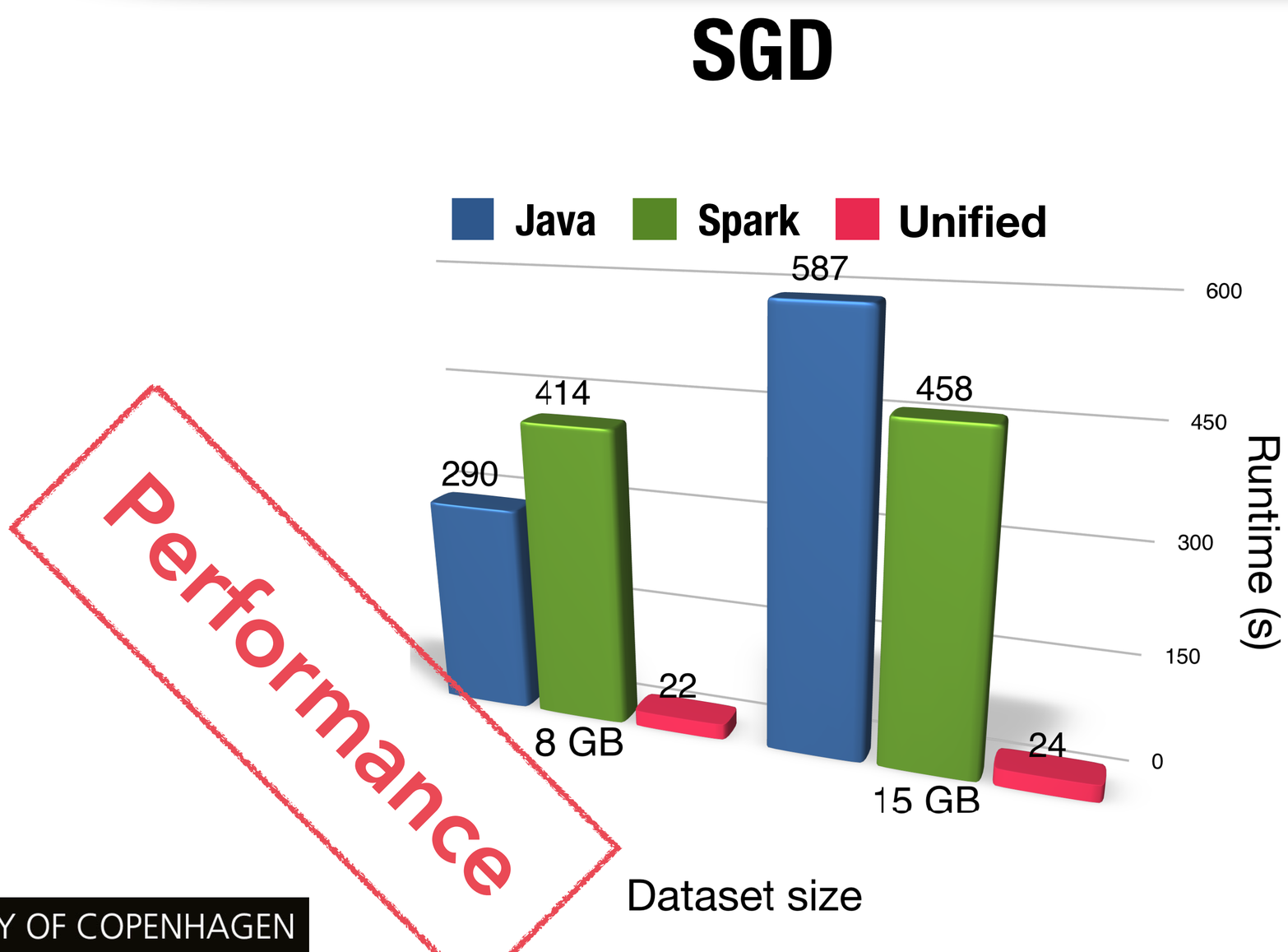
Cross-platform Data Analytics — Opportunistic

using ***several*** data processing platforms to reduce the cost of a query



Cross-platform Data Analytics — Opportunistic

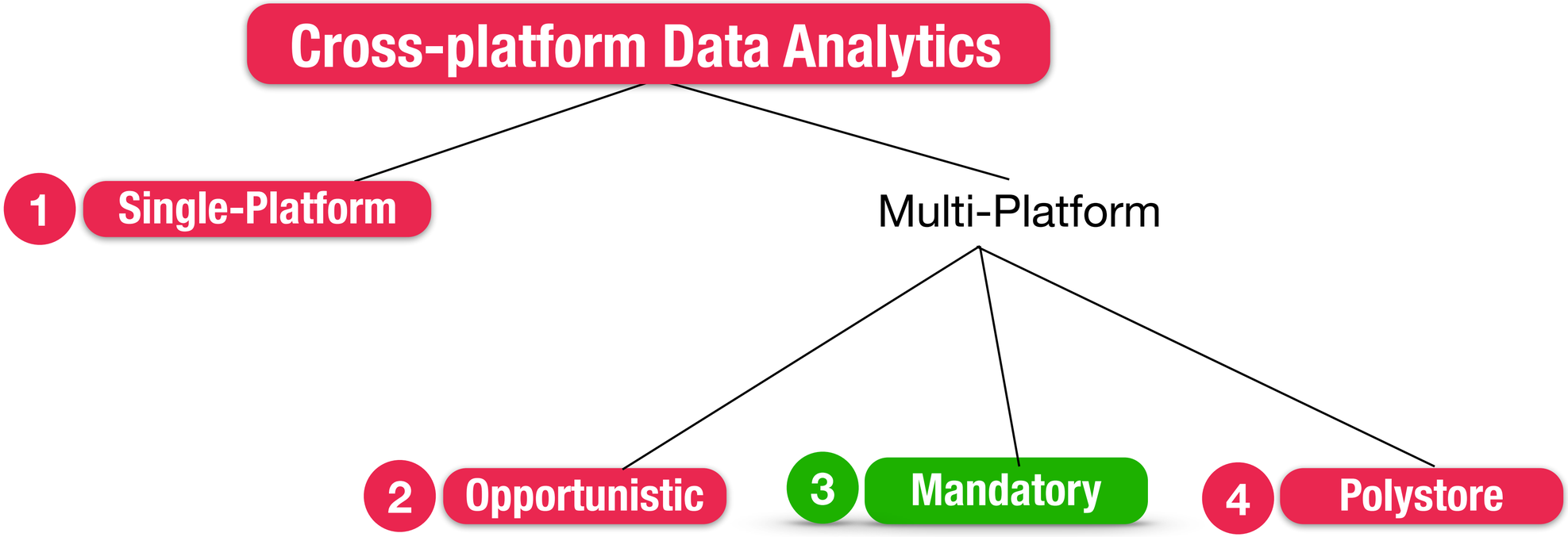
using ***several*** data processing platforms to reduce the cost of a query



System-Store Quadrant

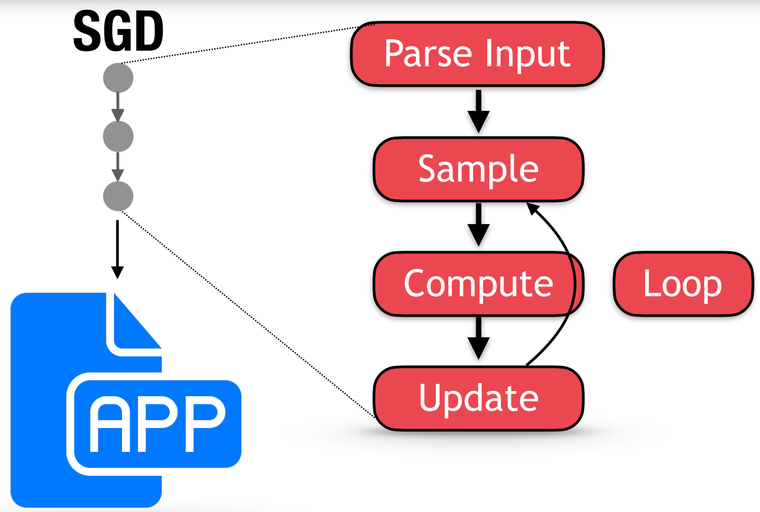
	Monoplatform	Polyplatform
Monostore	Single-Platform	Opportunistic
Polystore	Single-Platform	

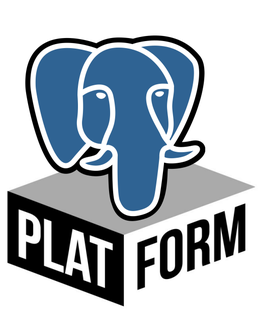
Cross-platform Data Processing Taxonomy

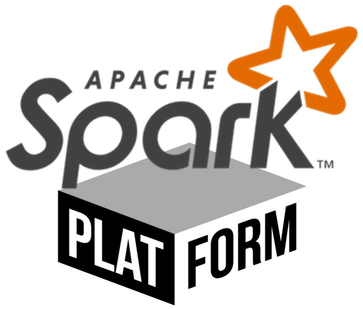


Cross-platform Data Analytics — Mandatory

using ***several*** data processing platforms to be able to process a query

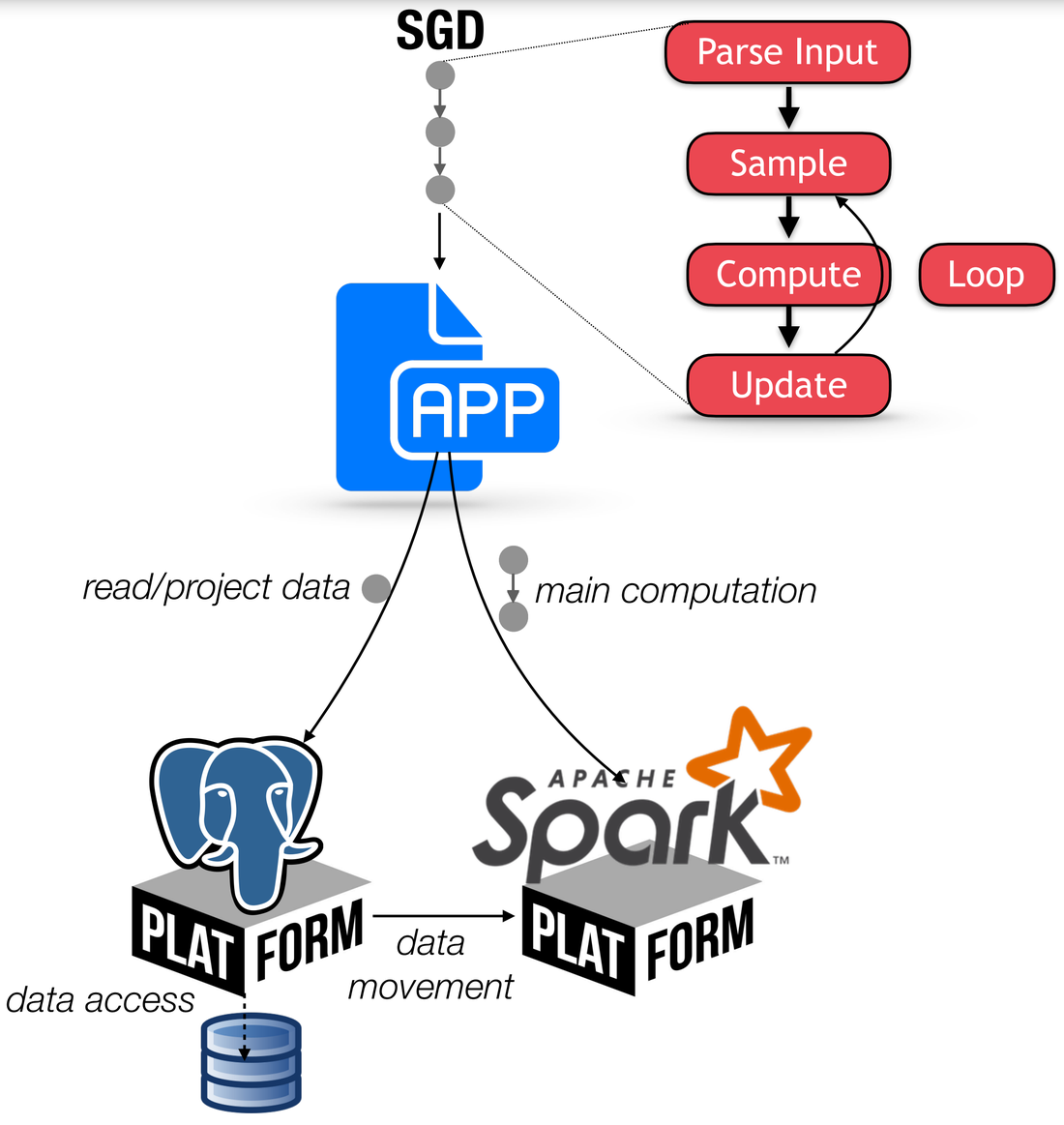






Cross-platform Data Analytics — Mandatory

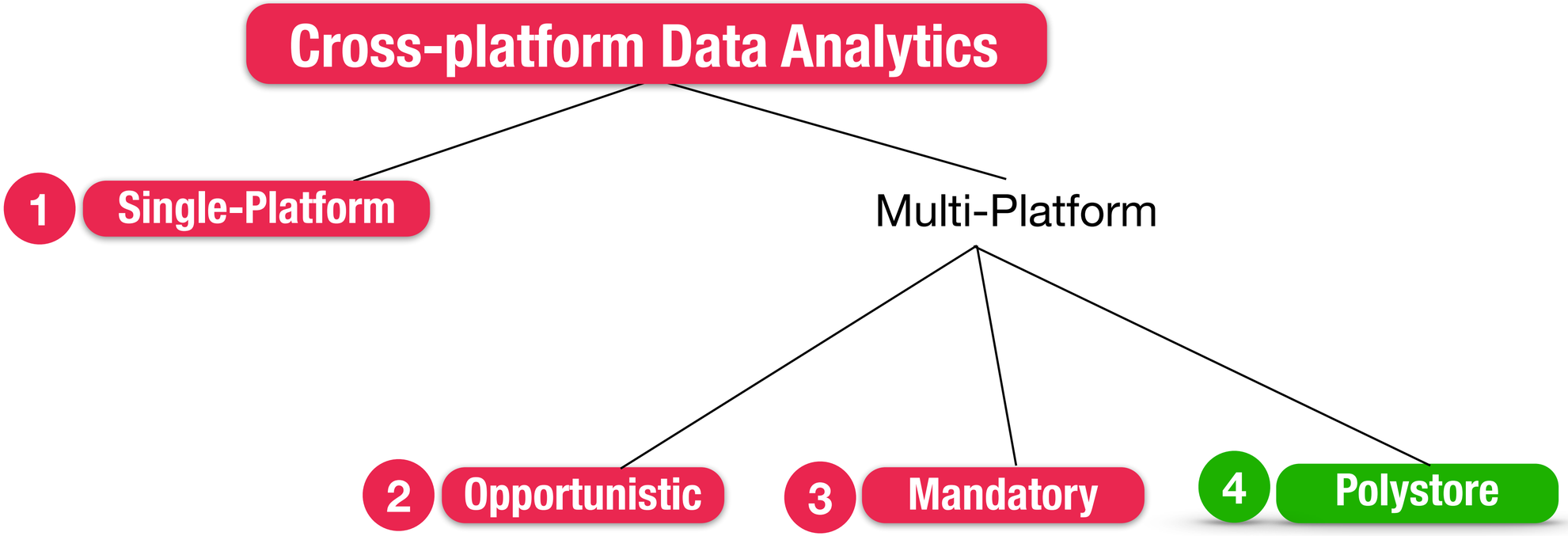
using ***several*** data processing platforms to be able to process a query



System-Store Quadrant

	Monoplatform	Polyplatform
Monostore	Single-Platform	Opportunistic Mandatory
Polystore	Single-Platform	

Cross-platform Data Processing Taxonomy

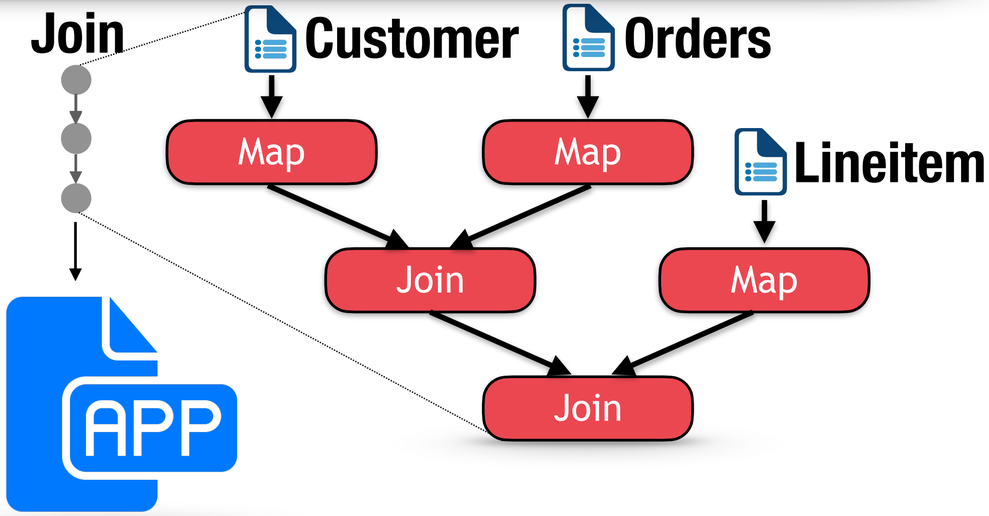


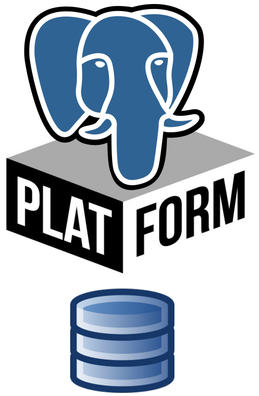
Cross-platform Data Analytics — Polystore

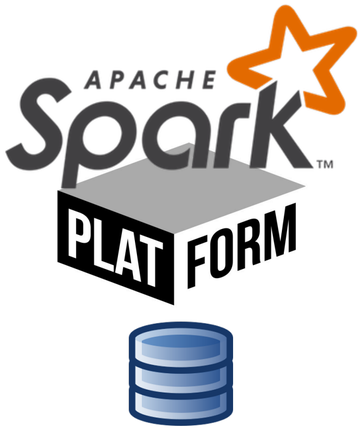
using ***several*** data processing platforms because data is spread

Cross-platform Data Analytics — Polystore

using ***several*** data processing platforms because data is spread

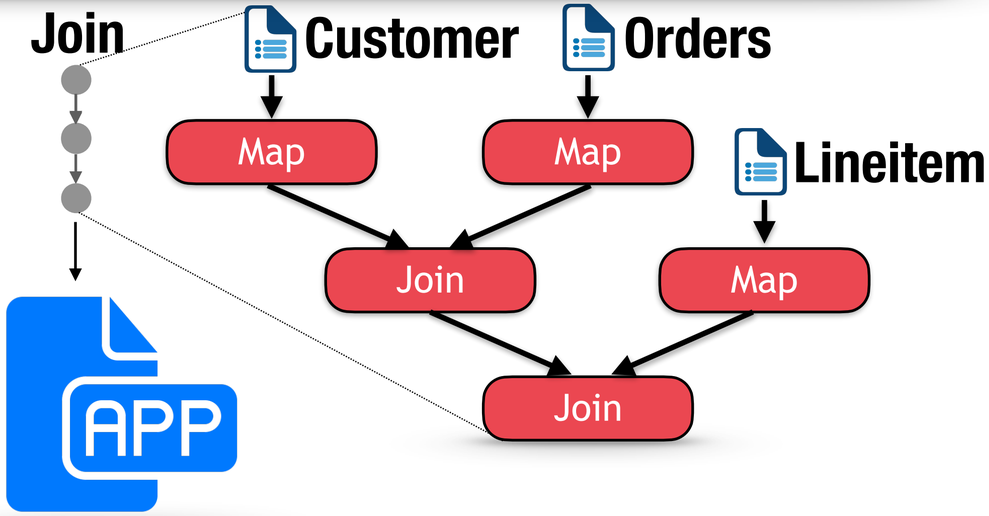


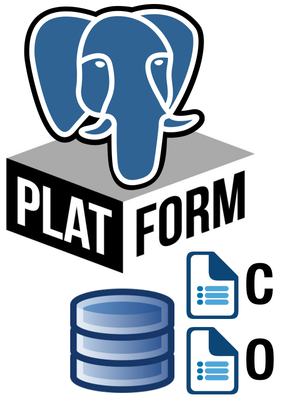


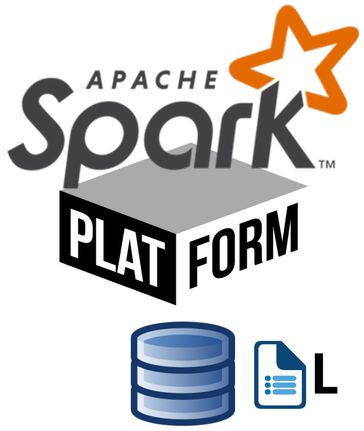


Cross-platform Data Analytics — Polystore

using ***several*** data processing platforms because data is spread

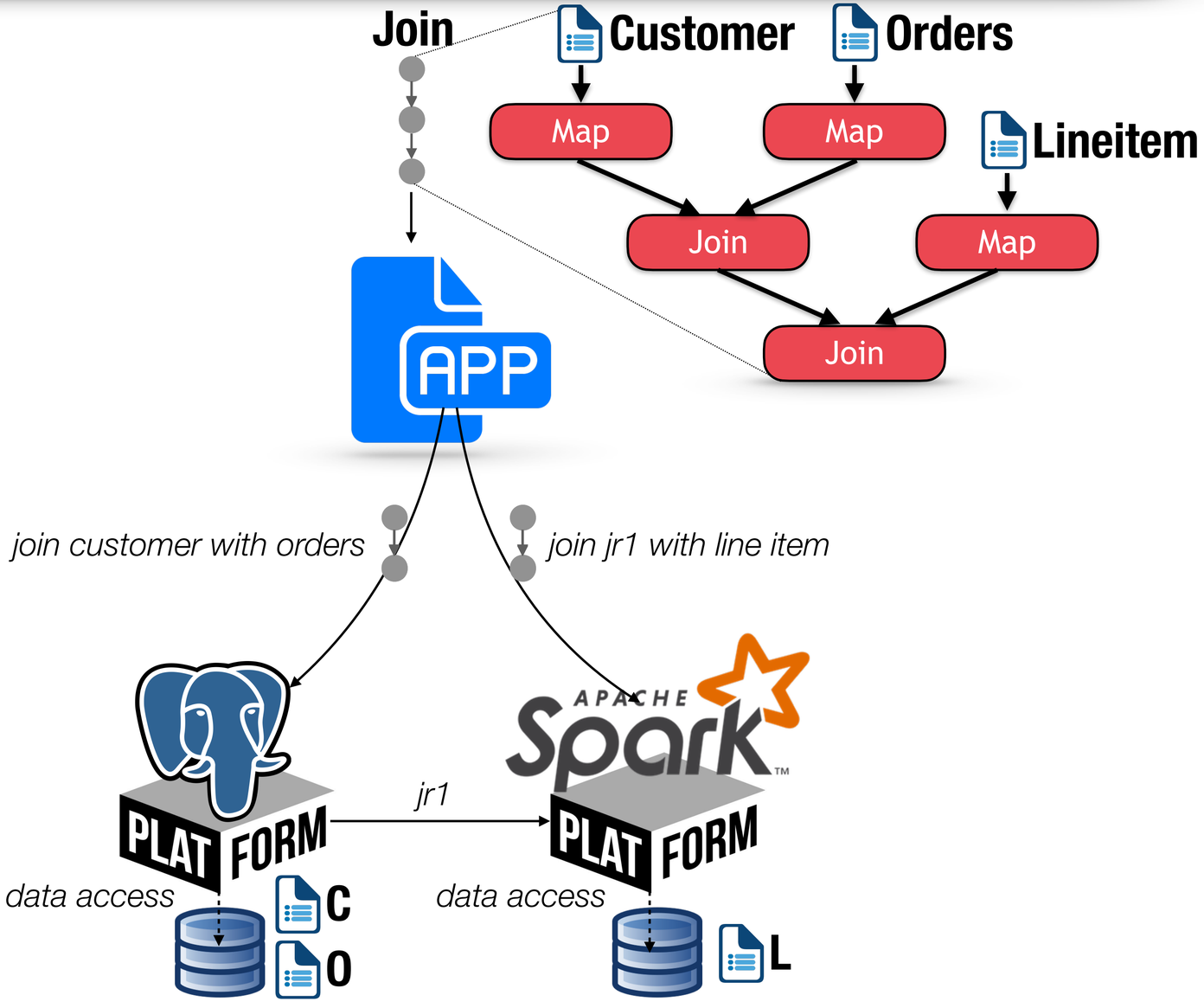






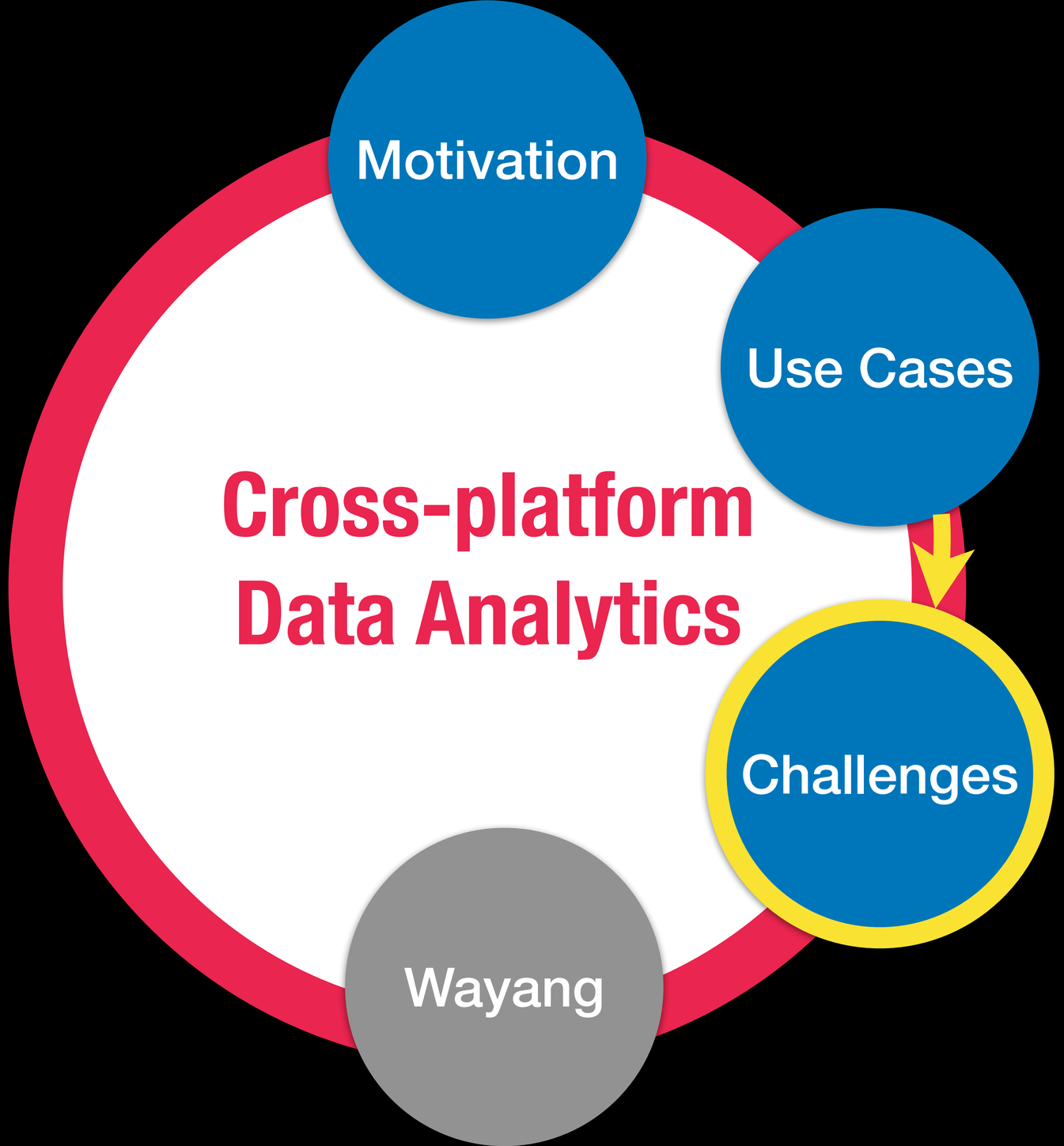
Cross-platform Data Analytics — Polystore

using ***several*** data processing platforms because data is spread



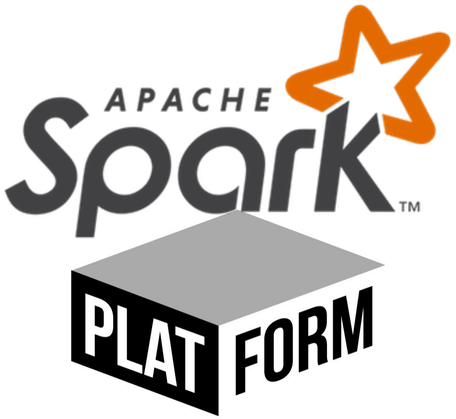
System-Store Quadrant

	Monoplatform	Polyplatform
Monostore	**Single-Platform**	**Opportunistic** **Mandatory**
Polystore	**Single-Platform**	**Polystore**


Motivation
Use Cases
Cross-platform
Data Analytics
Challenges
Wayang

Challenges

Challenges

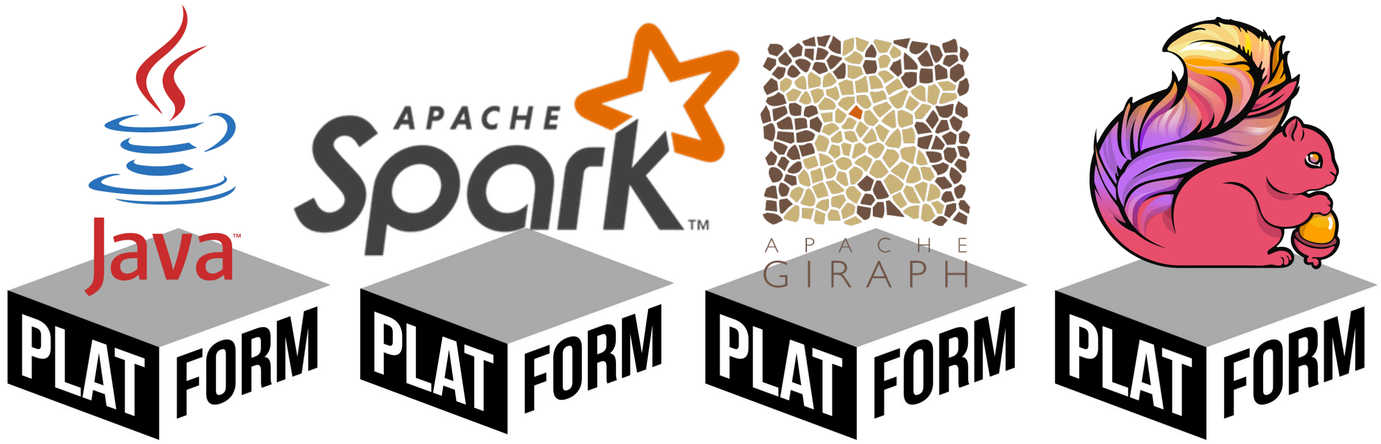


Challenges

1 **Decoupling Applications**

Challenges

1 Decoupling Applications



Challenges

1 Decoupling Applications

2 Data Movement

Challenges

1 Decoupling Applications

3

Automatic Cross-platform Data Analytics

2 Data Movement

Challenges

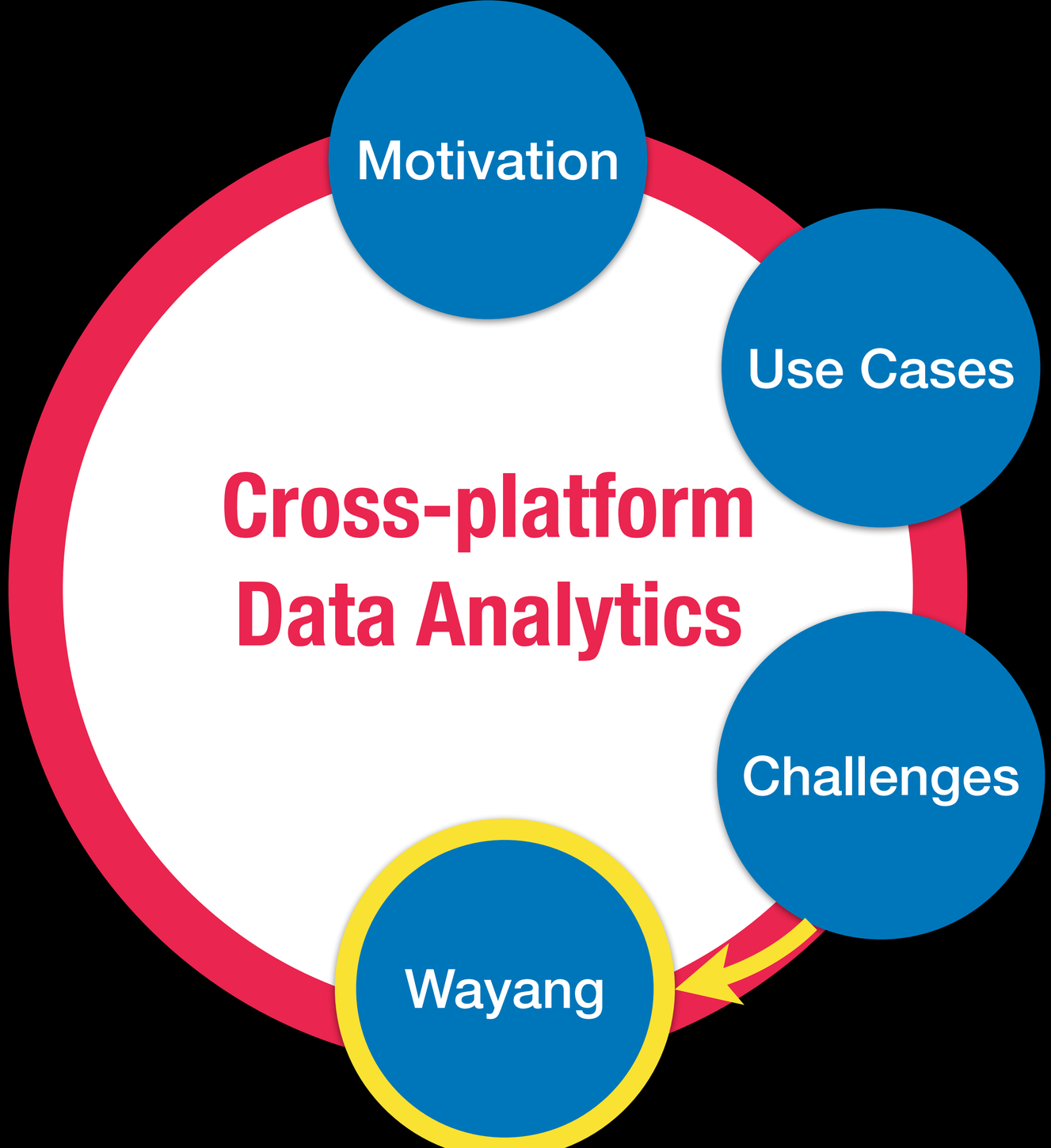
1 Decoupling Applications

4 Extensibility

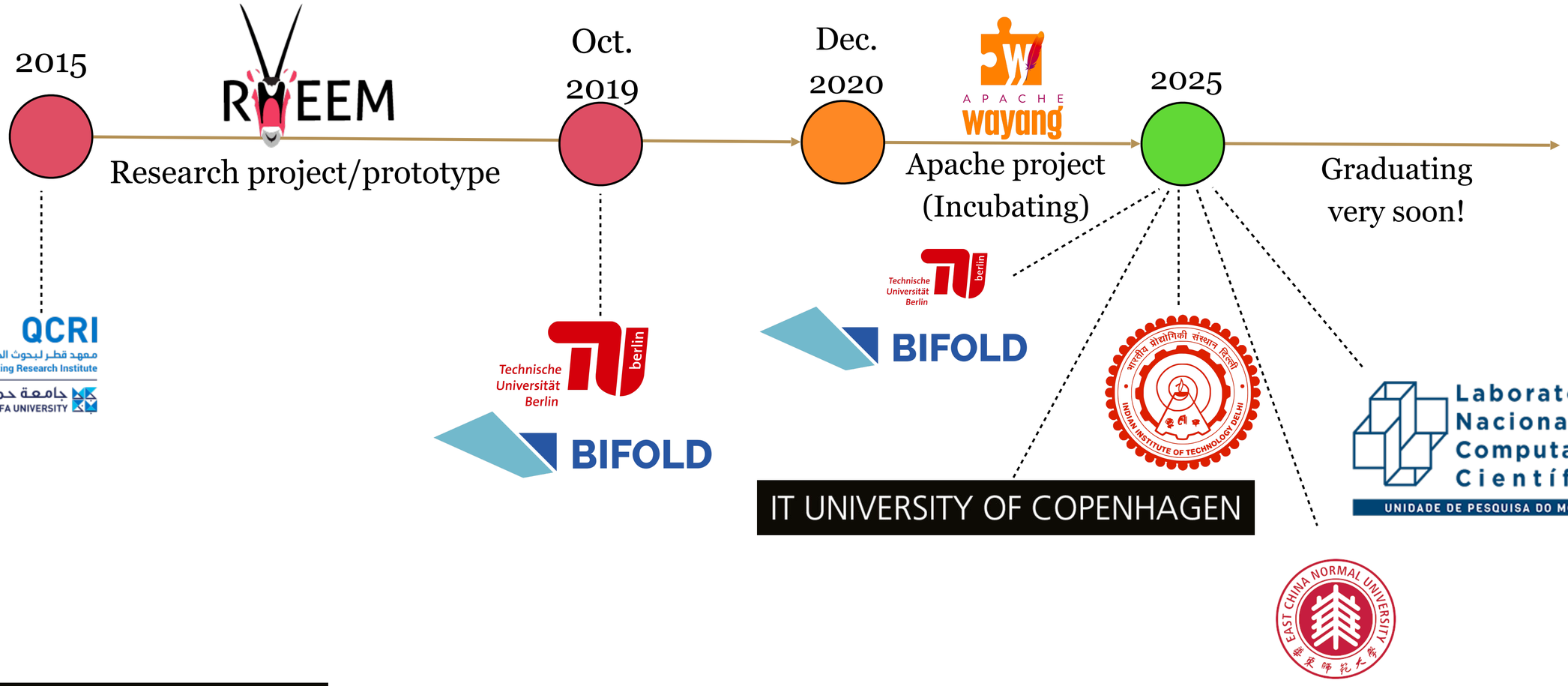
3

Automatic Cross-platform Data Analytics

2 Data Movement


Motivation
Use Cases
Cross-platform
Data Analytics
Challenges
Wayang

Apache Wayang History



Idea Behind Apache Wayang

Applications

APACHE wayang

- Optimization & Coordination -

Apache Wayang

Data Processing Engines

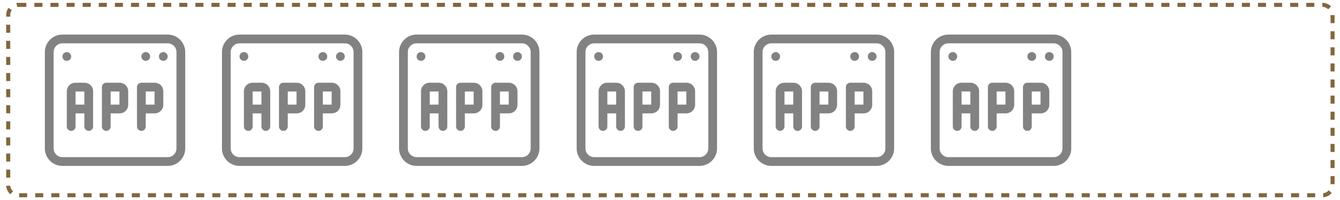
- Data Processing -

Data Sources

- Data Extraction -

Idea Behind Apache Wayang

Applications



- Optimization & Coordination -

Data Processing Engines

- Data Processing -

Data Sources

- Data Extraction -

Example: Platform independence/agnosticity

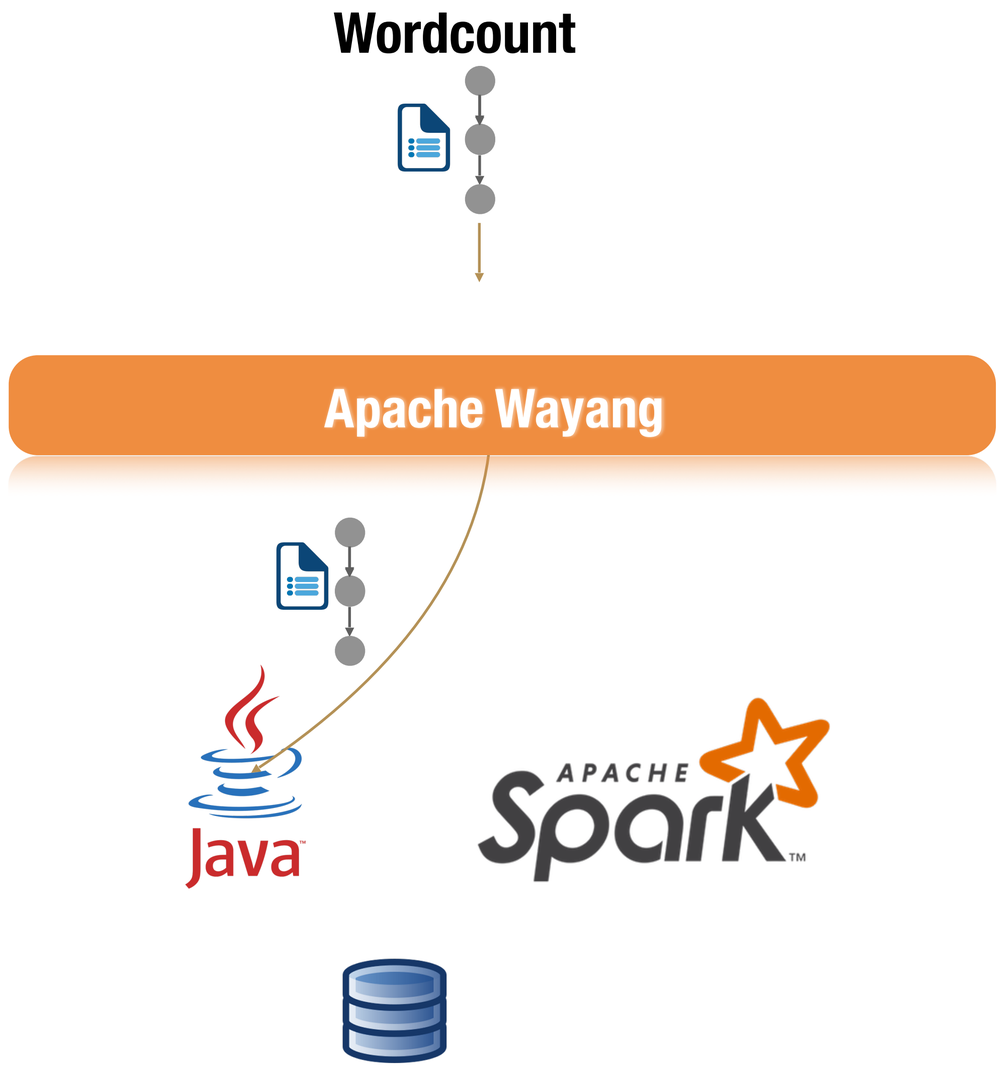
using ***any single*** data processing platform to process a query

Wordcount

Apache Wayang

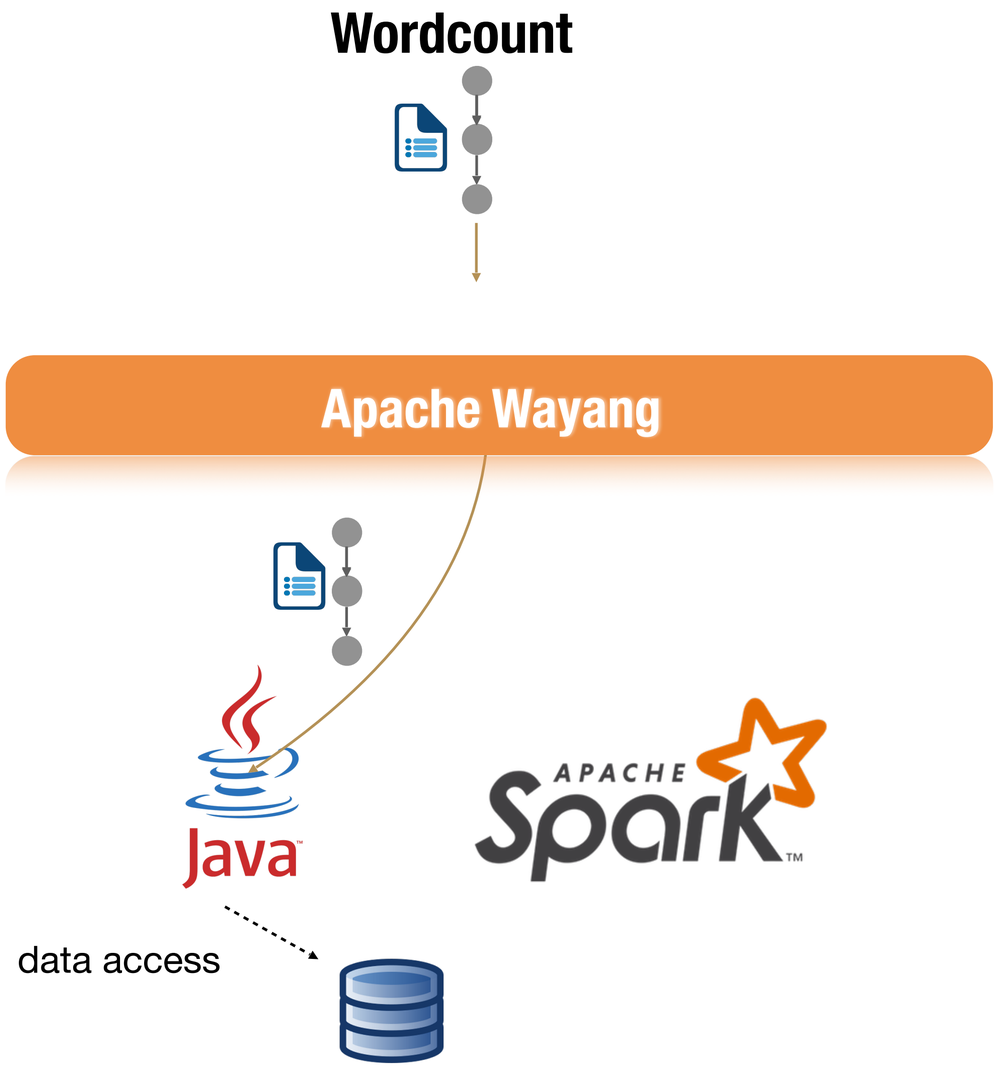
Example: Platform independence/agnosticity

using ***any single*** data processing platform to process a query



Example: Platform independence/agnosticity

using ***any single*** data processing platform to process a query



Example: Pla

using ***any single*** data

```
import org.apache.wayang.api._
import org.apache.wayang.core.api.{Configuration, WayangContext}
import org.apache.wayang.java.Java
import java.io.File;

object Wordcount {

 def main(args: Array[String]): Unit = {
   val inputFile = new File( pathname = "book.txt").toURI().toString()

   val context = new WayangContext()
     .withPlugin(Java.basicPlugin)
   var result = wordcount(inputFile, context)

 }

 def wordcount(inputFile:String, context: WayangContext): Iterable[(String, Int)] = {
   val planBuilder = new PlanBuilder(context)

   planBuilder                                                    : PlanBuilder
     .withJobName( jobName = s"WordCount ($inputFile)")           : PlanBuilder
     .readTextFile(inputFile)                                     : DataQuanta[String]
     .flatMap(_.split( regex = "\\W+"))                           : DataQuanta[String]
     .filter(_.nonEmpty)                                          : DataQuanta[String]
     .map(word => (word.toLowerCase, 1))                          : DataQuanta[(String, Int)]
     .reduceByKey(_._1, (c1, c2) => (c1._1, c1._2 + c2._2))       : DataQuanta[(String, Int)]
     .collect()                                                   : Iterable[(String, Int)]
 }
}
```

Example: Pla

using ***any single*** data

```
import org.apache.wayang.api._
import org.apache.wayang.core.api.{Configuration, WayangContext}
import org.apache.wayang.java.Java
import java.io.File;

object Wordcount {

 def main(args: Array[String]): Unit = {
   val inputFile = new File( pathname = "book.txt").toURI().toString()

   val context = new WayangContext()
     .withPlugin(Java.basicPlugin)
   var result = wordcount(inputFile, context)

 }

 def wordcount(inputFile:String, context: WayangContext): Iterable[(String, Int)] = {
   val planBuilder = new PlanBuilder(context)

   planBuilder                                                    : PlanBuilder
     .withJobName( jobName = s"WordCount ($inputFile)")           : PlanBuilder
     .readTextFile(inputFile)                                     : DataQuanta[String]
     .flatMap(_.split( regex = "\\W+"))                           : DataQuanta[String]
     .filter(_.nonEmpty)                                          : DataQuanta[String]
     .map(word => (word.toLowerCase, 1))                          : DataQuanta[(String, Int)]
     .reduceByKey(_._1, (c1, c2) => (c1._1, c1._2 + c2._2))       : DataQuanta[(String, Int)]
     .collect()                                                   : Iterable[(String, Int)]
 }
}
```

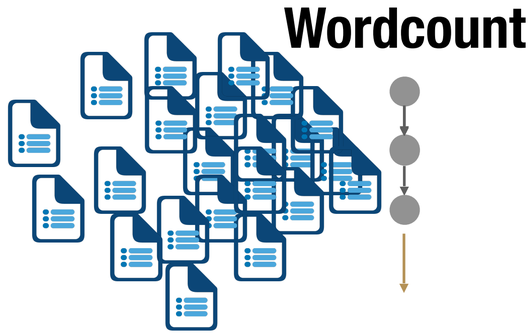
Example: Pla

using ***any single*** data

```
import org.apache.wayang.api._
import org.apache.wayang.core.api.{Configuration, WayangContext}
import org.apache.wayang.java.Java
import java.io.File;

object Wordcount {

 def main(args: Array[String]): Unit = {
   val inputFile = new File( pathname = "book.txt").toURI().toString()

   val context = new WayangContext()
     .withPlugin(Java.basicPlugin)
   var result = wordcount(inputFile, context)

 }

 def wordcount(inputFile:String, context: WayangContext): Iterable[(String, Int)] = {
   val planBuilder = new PlanBuilder(context)

   planBuilder                                                    : PlanBuilder
     .withJobName( jobName = s"WordCount ($inputFile)")           : PlanBuilder
     .readTextFile(inputFile)                                     : DataQuanta[String]
     .flatMap(_.split( regex = "\\W+"))                           : DataQuanta[String]
     .filter(_.nonEmpty)                                          : DataQuanta[String]
     .map(word => (word.toLowerCase, 1))                          : DataQuanta[(String, Int)]
     .reduceByKey(_._1, (c1, c2) => (c1._1, c1._2 + c2._2))       : DataQuanta[(String, Int)]
     .collect()                                                   : Iterable[(String, Int)]
 }
}
```

Example: Platform independence/agnosticity

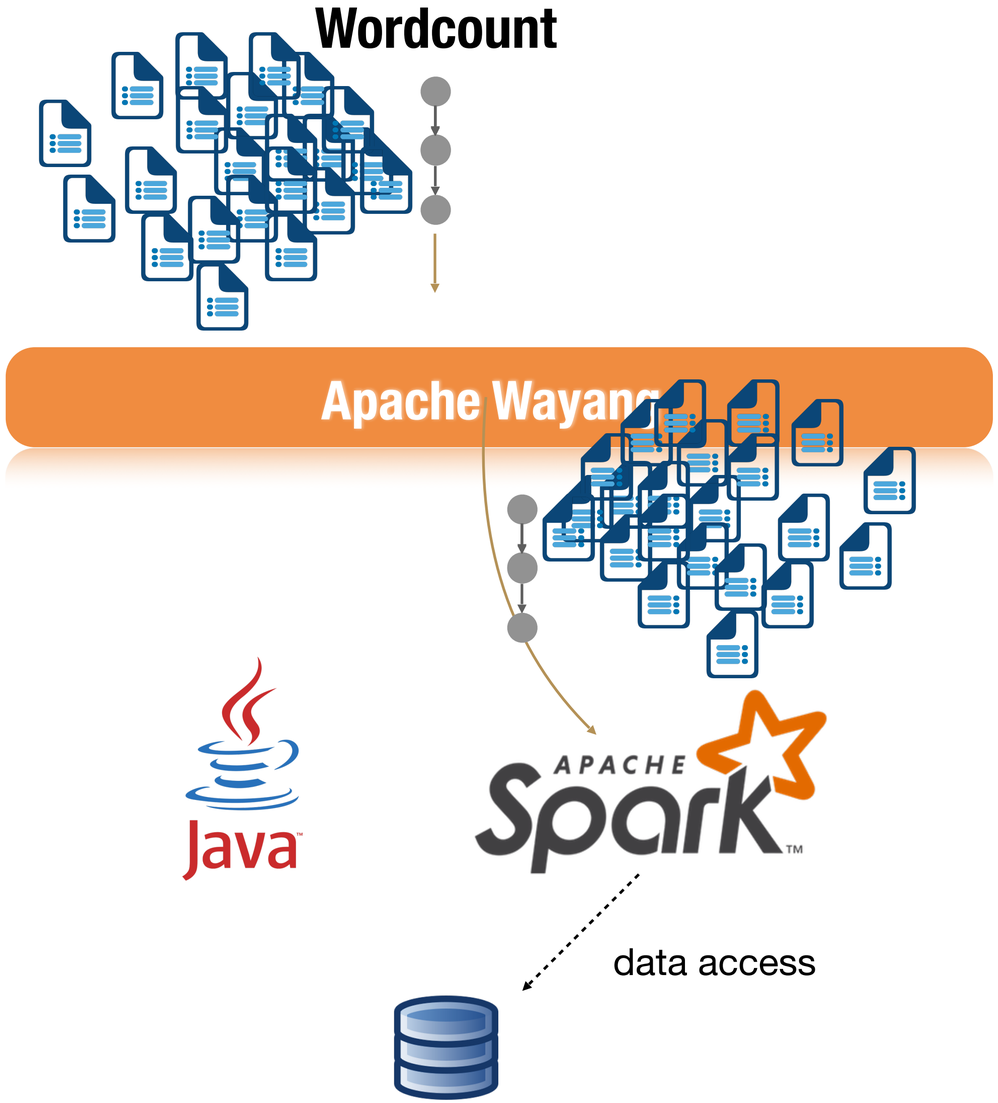
using ***any single*** data processing platform to process a query



Apache Wayang

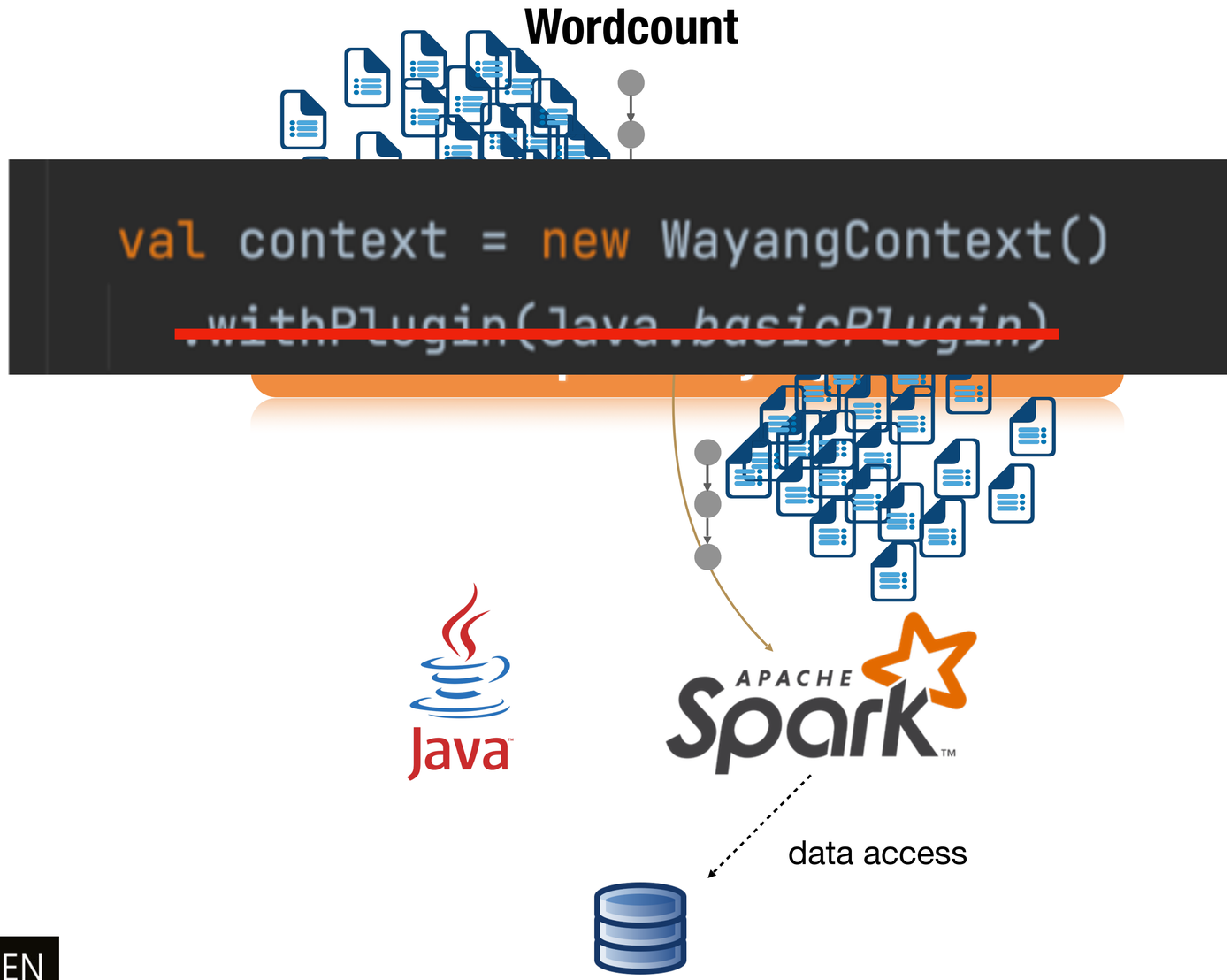
Example: Platform independence/agnosticity

using ***any single*** data processing platform to process a query



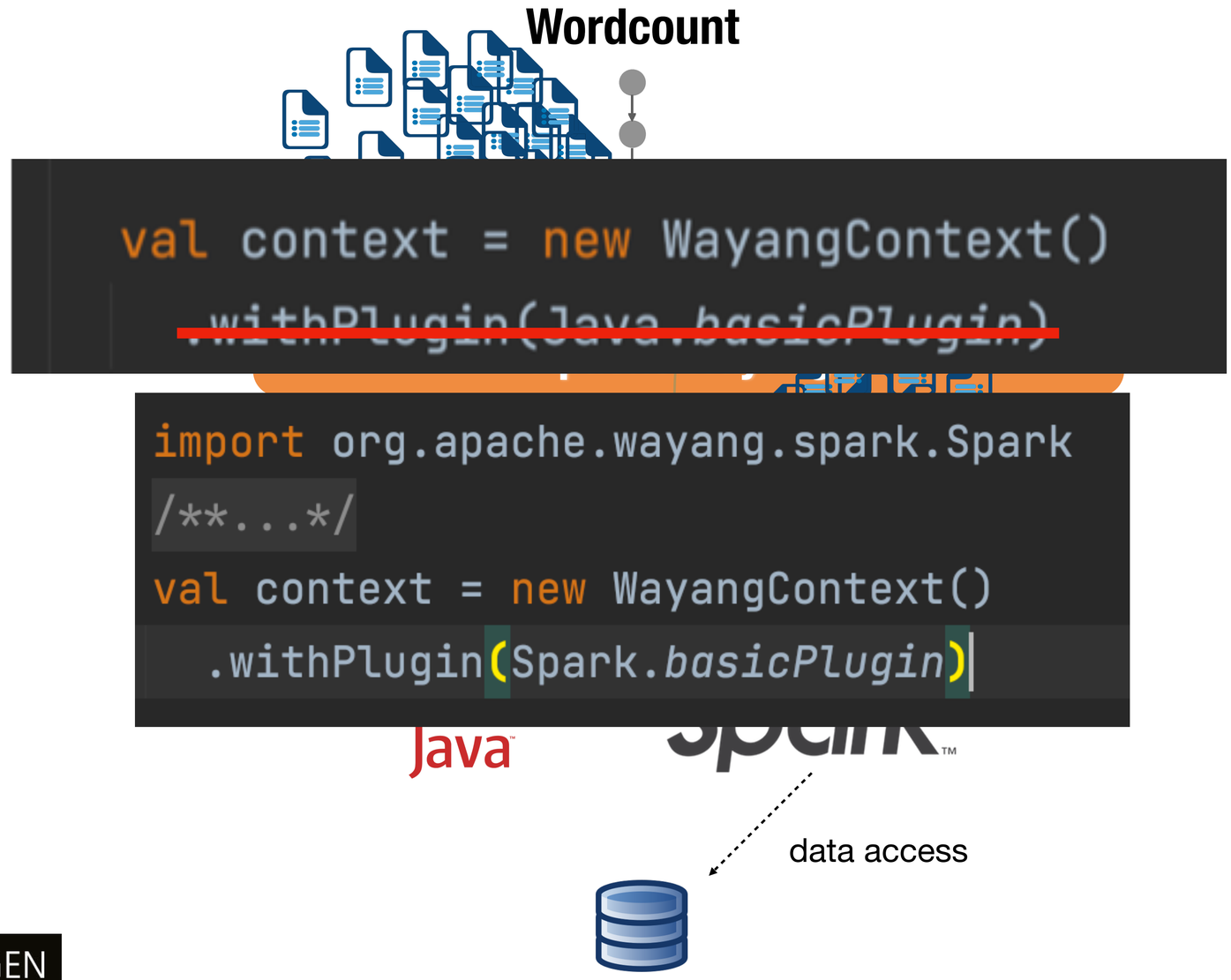
Example: Platform independence/agnosticity

using ***any single*** data processing platform to process a query



Example: Platform independence/agnosticity

using ***any single*** data processing platform to process a query



Example: Platform independence/agnosticity

using ***any single*** data processing platform to process a query

Example: Platform independence/agnosticity

using ***any single*** data processing platform to process a query

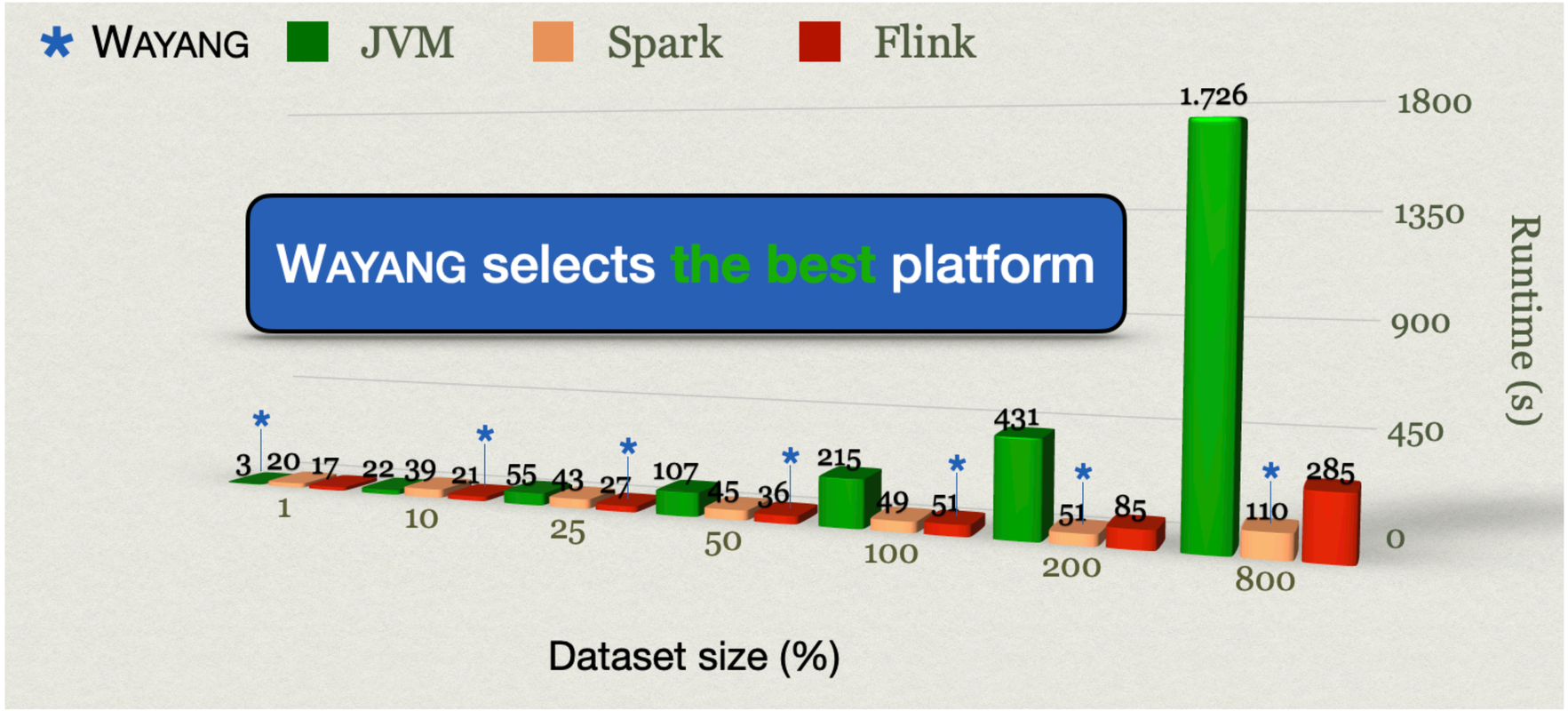
```
import org.apache.wayang.java.Java
import org.apache.wayang.spark.Spark
import org.apache.wayang.flink.Flink
/**...*/
val context = new WayangContext()
  .withPlugin(Java.basicPlugin)
  .withPlugin(Spark.basicPlugin)
  .withPlugin(Flink.basicPlugin)
```

Example: Platform independence/agnosticity

using ***any single*** data processing platform to process a query

Query: WordCount

Datasets: `Wikipedia`

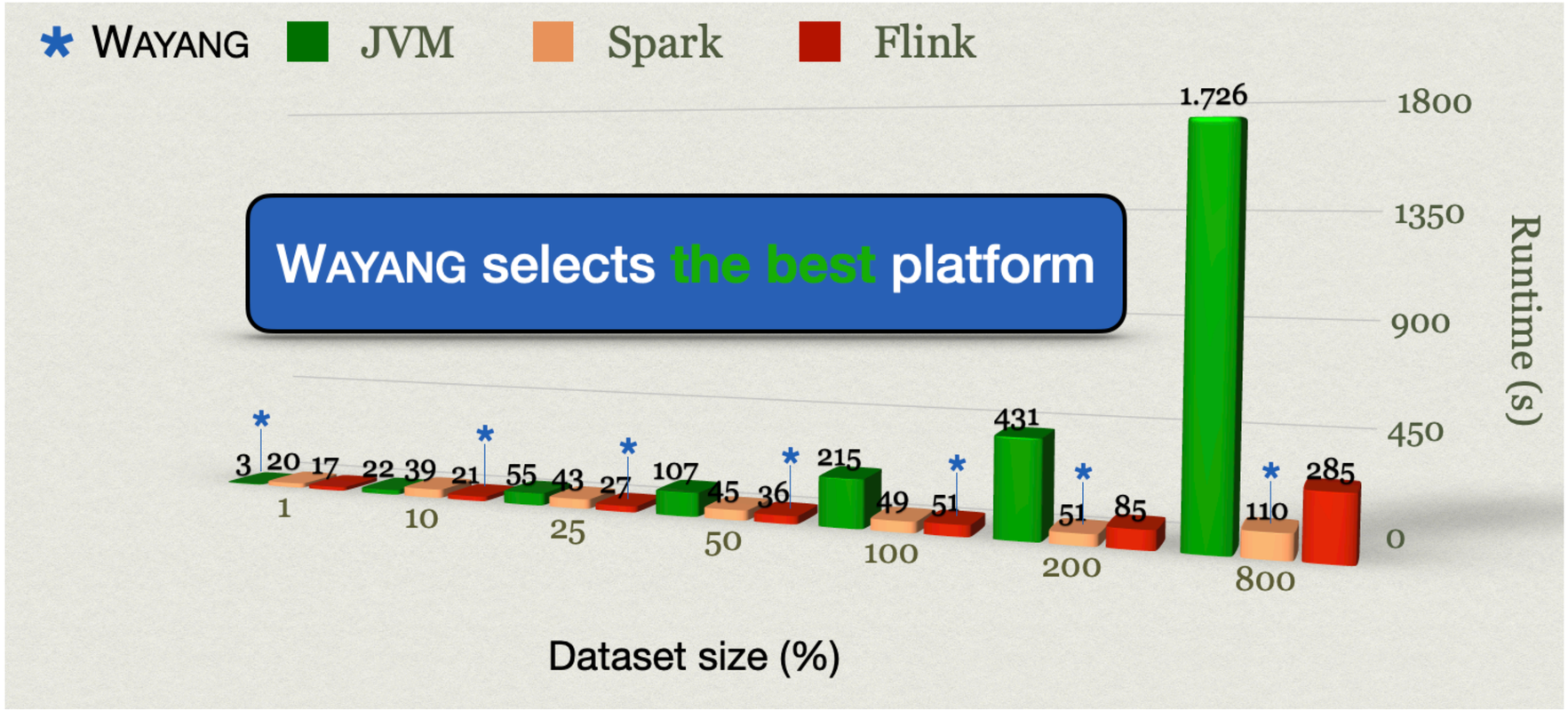


Example: Platform independence/agnosticity

using ***any single*** data processing platform to process a query

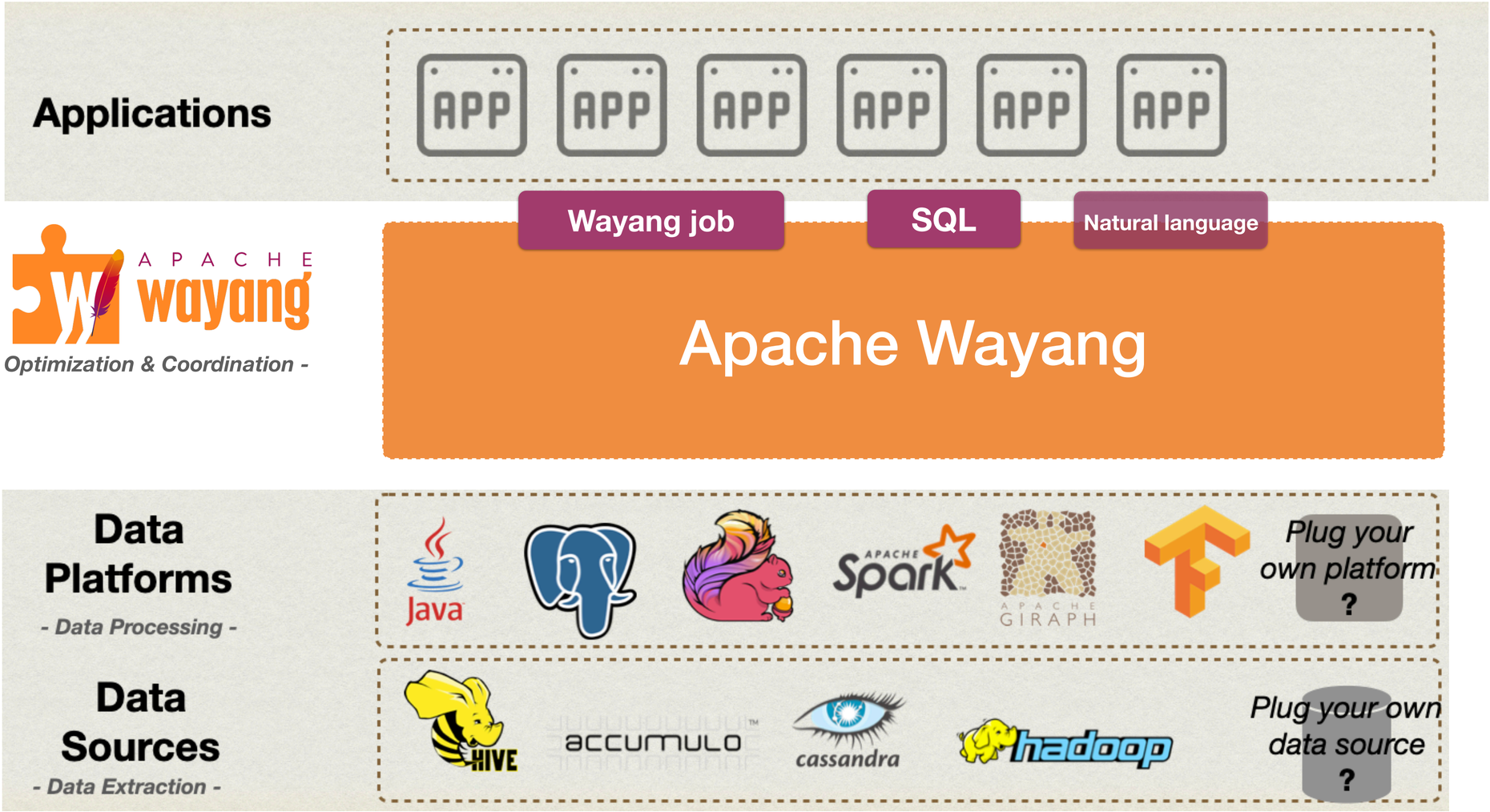
Query: WordCount

Datasets: `Wikipedia`

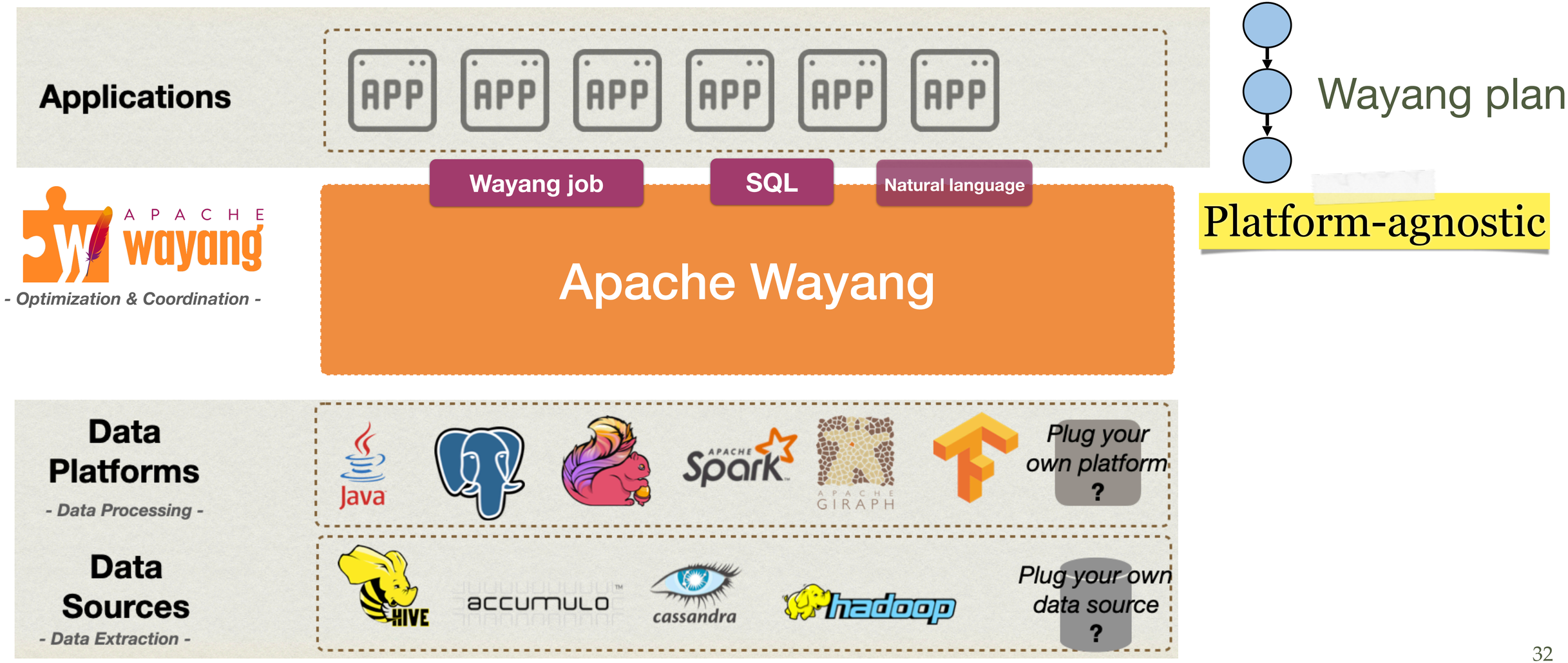


- Runtime speedup
- Cost savings
- Energy savings
- Easy platform migration

How does Apache Wayang work?



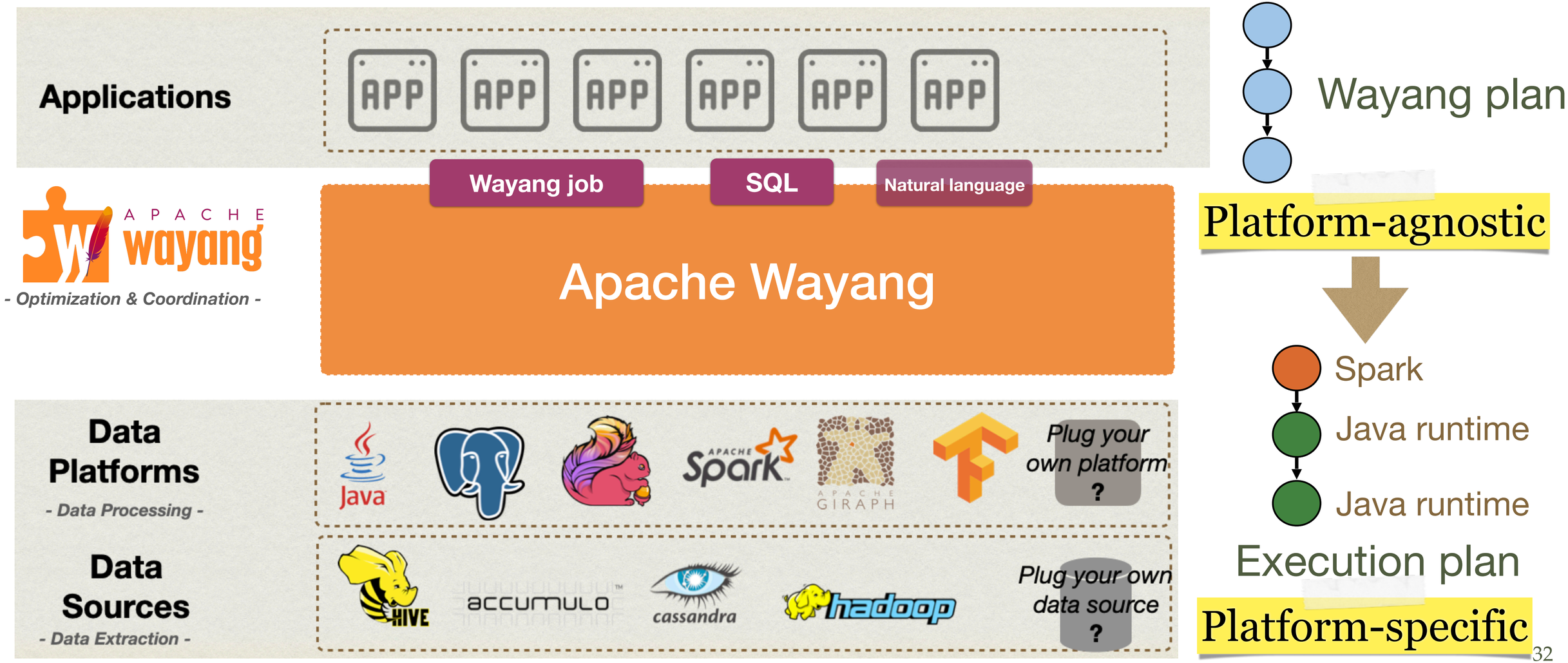
How does Apache Wayang work?



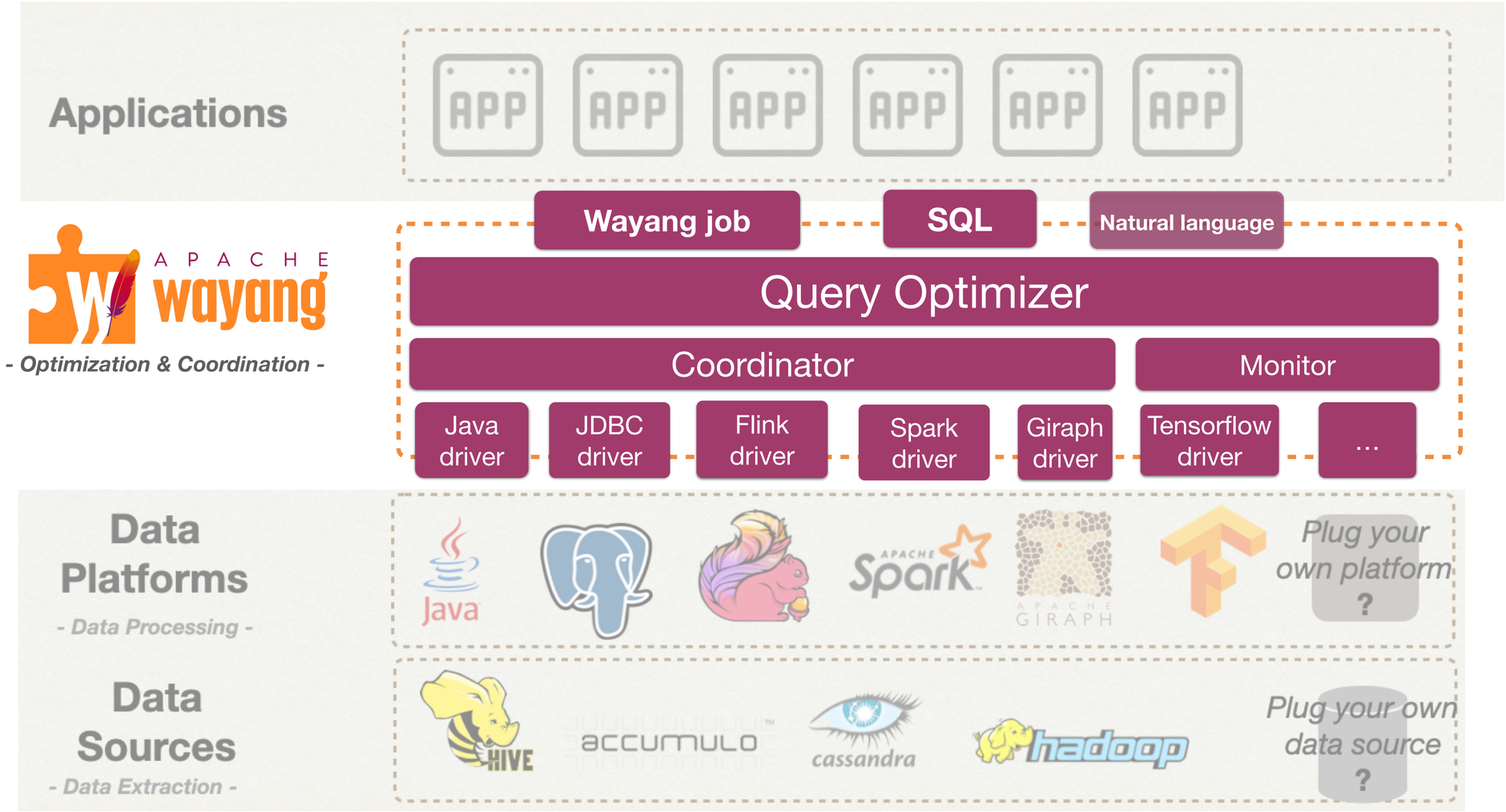
Wayang plan

Platform-agnostic

How does Apache Wayang work?



Apache Wayang Architecture



Challenges

1 Decoupling Applications

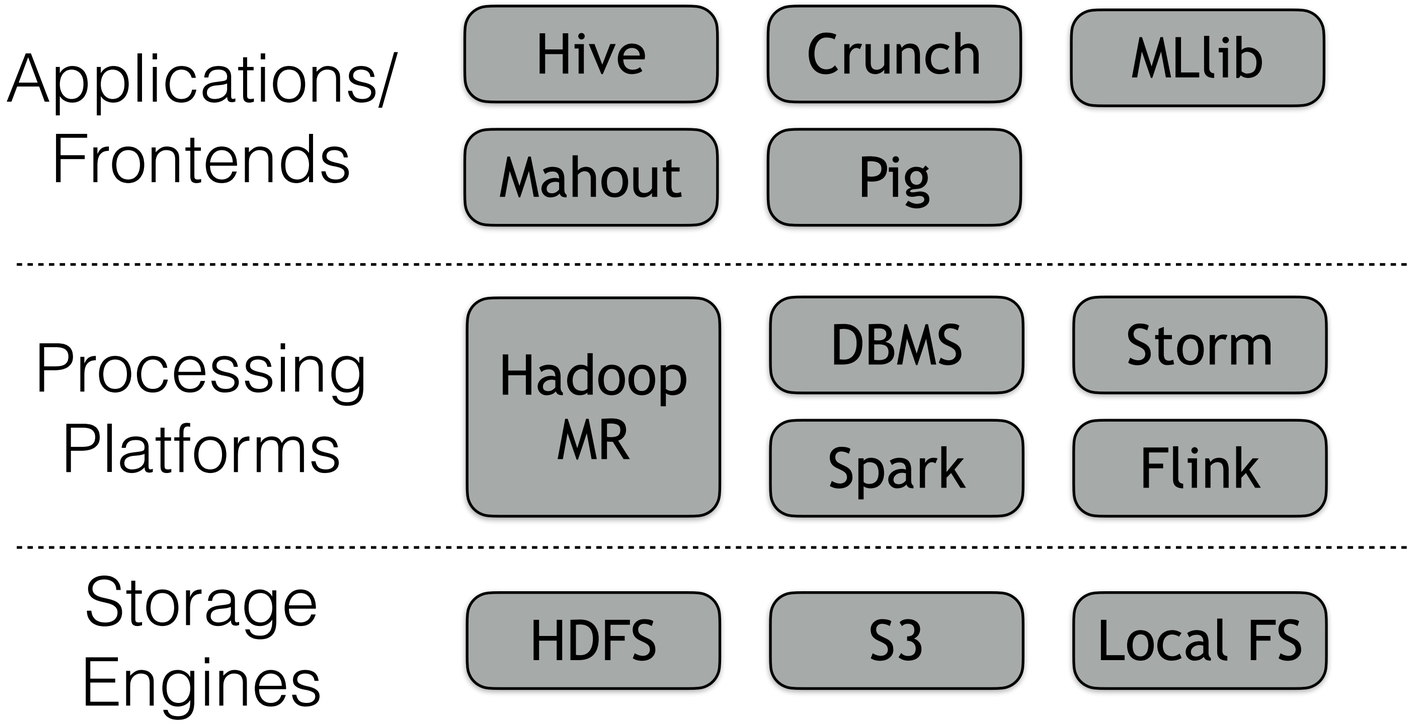
4 Extensibility

3

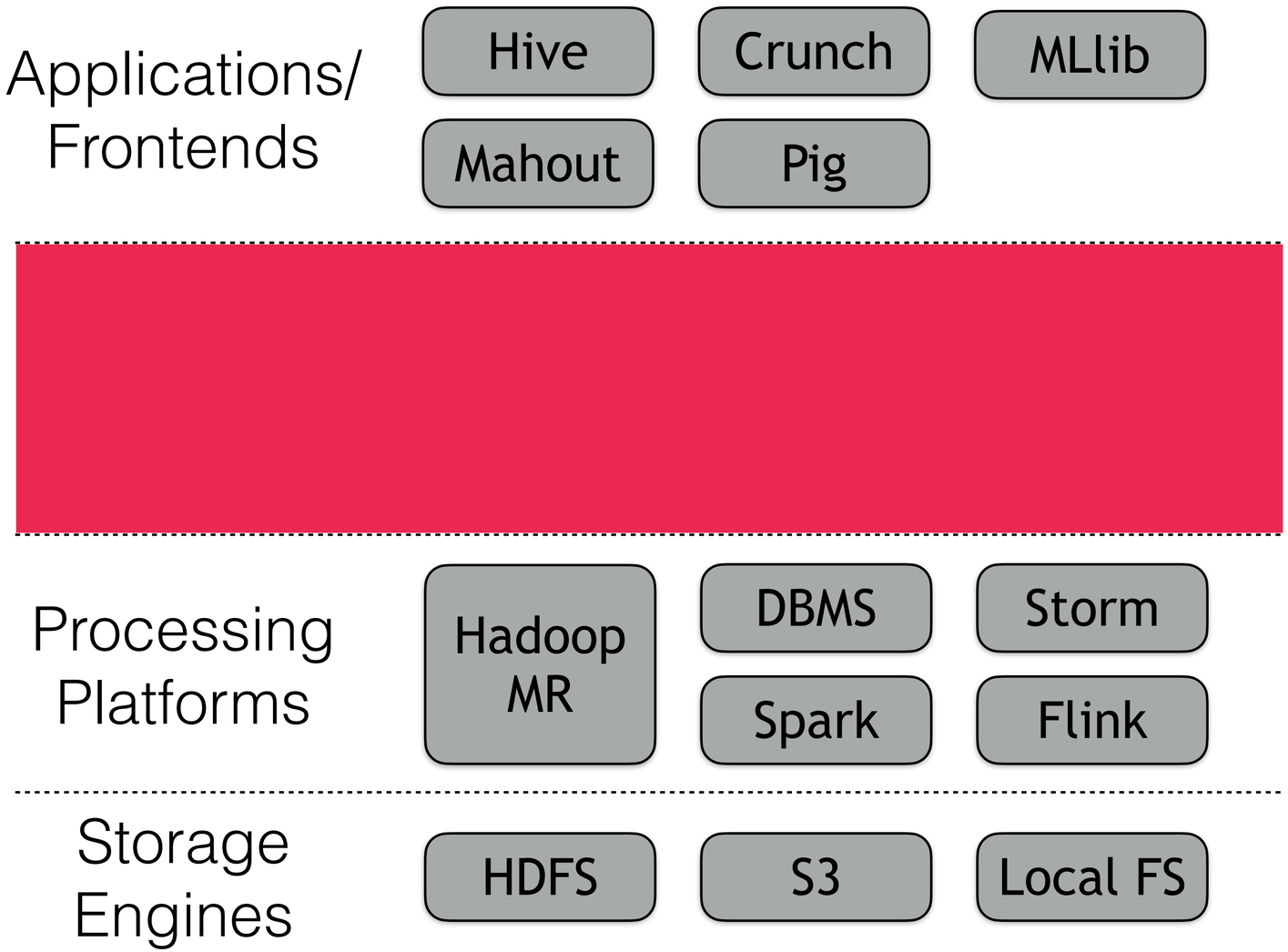
Automatic Cross-platform Data Analytics

2 Data Movement

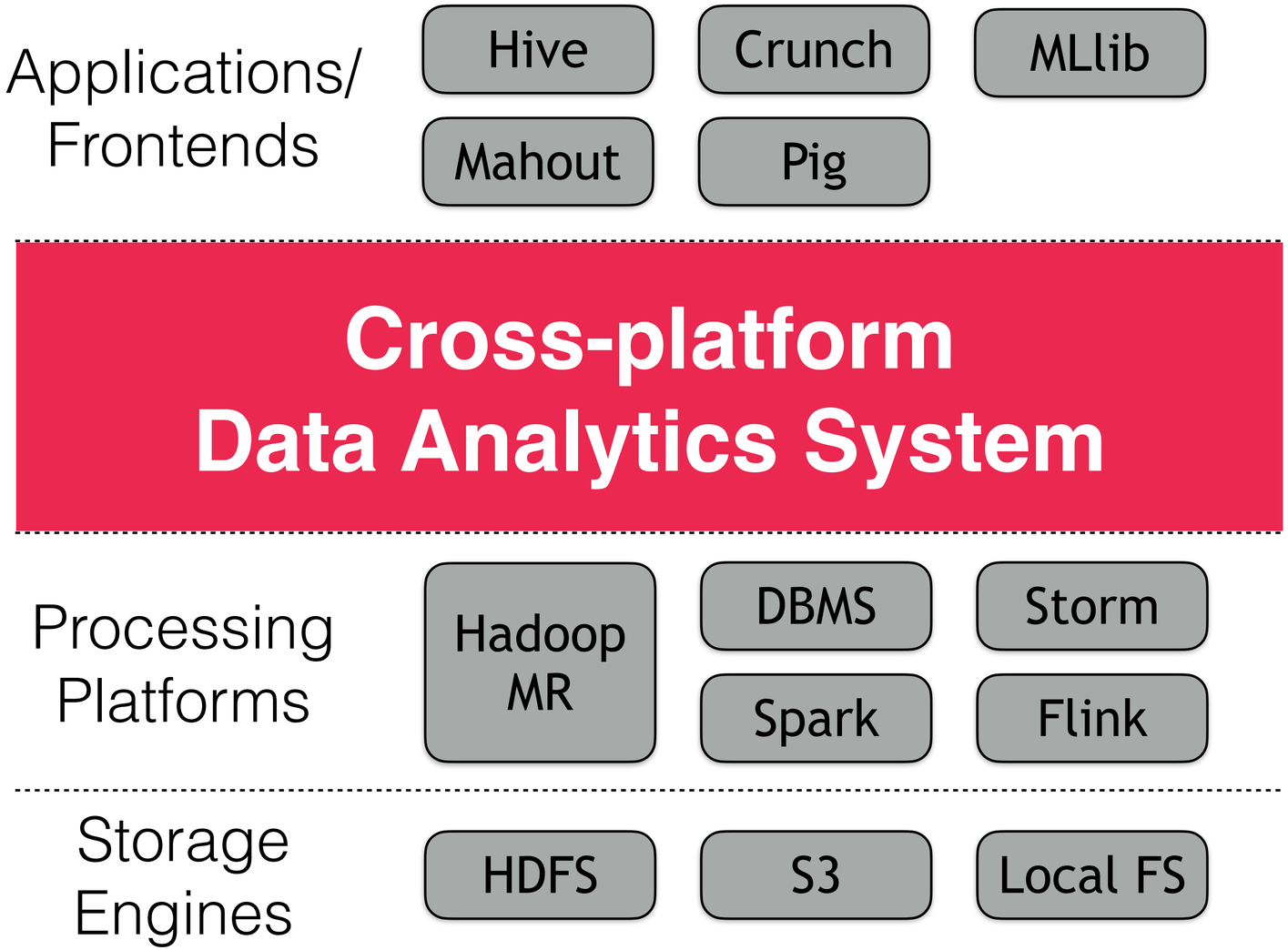
Decoupling Applications



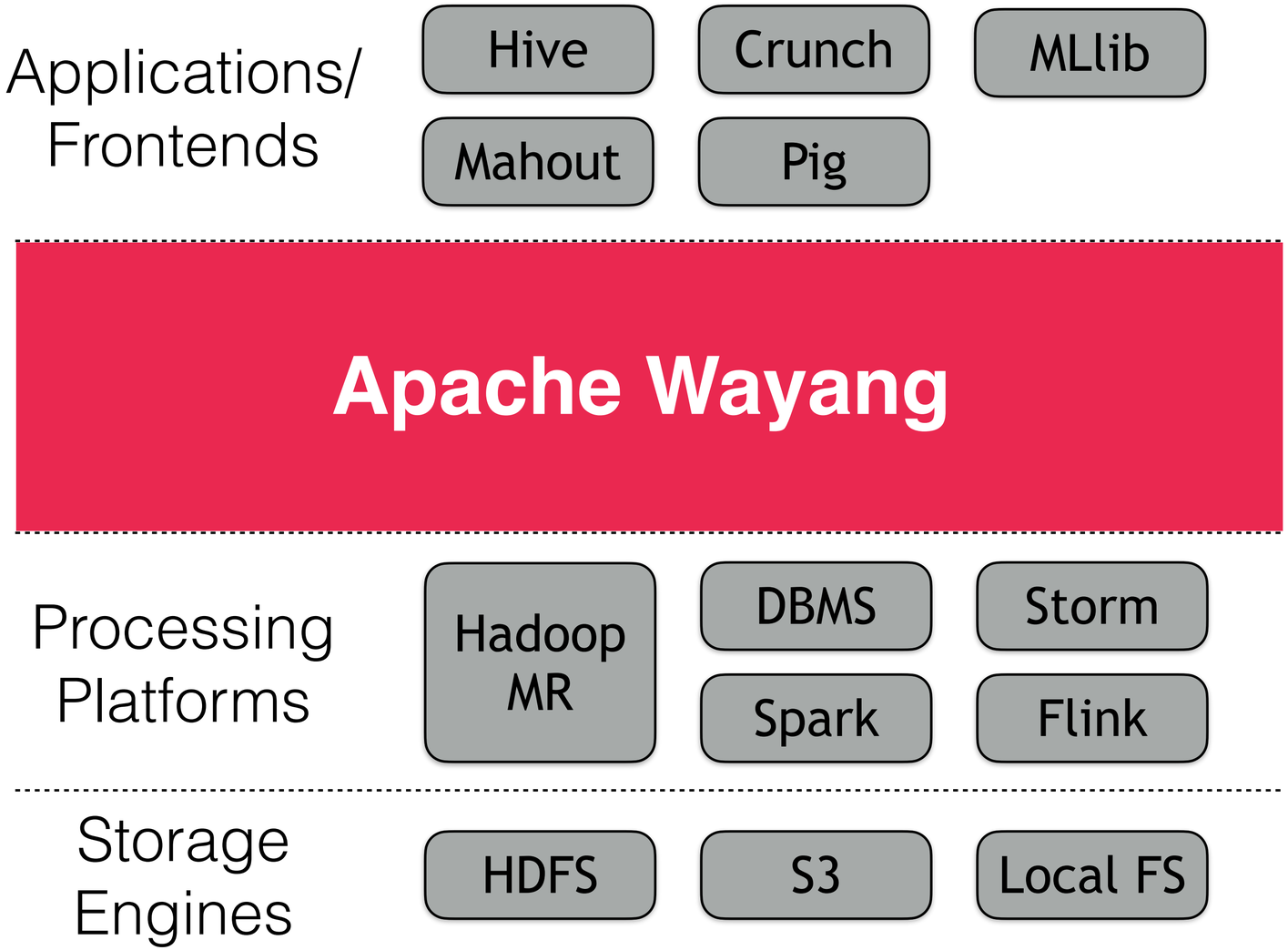
Decoupling Applications



Decoupling Applications



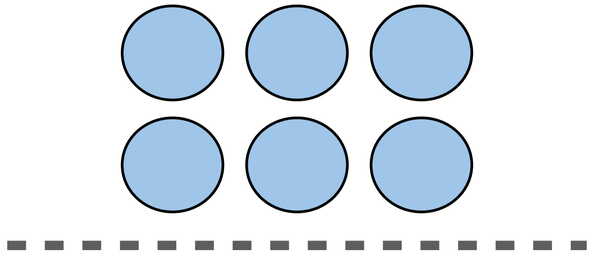
Decoupling Applications



Apache Wayang

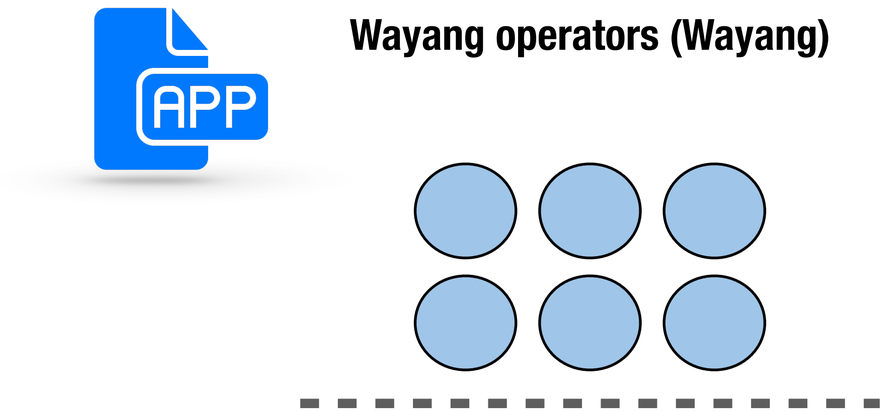
2 layers of operators

Wayang operators (Wayang)



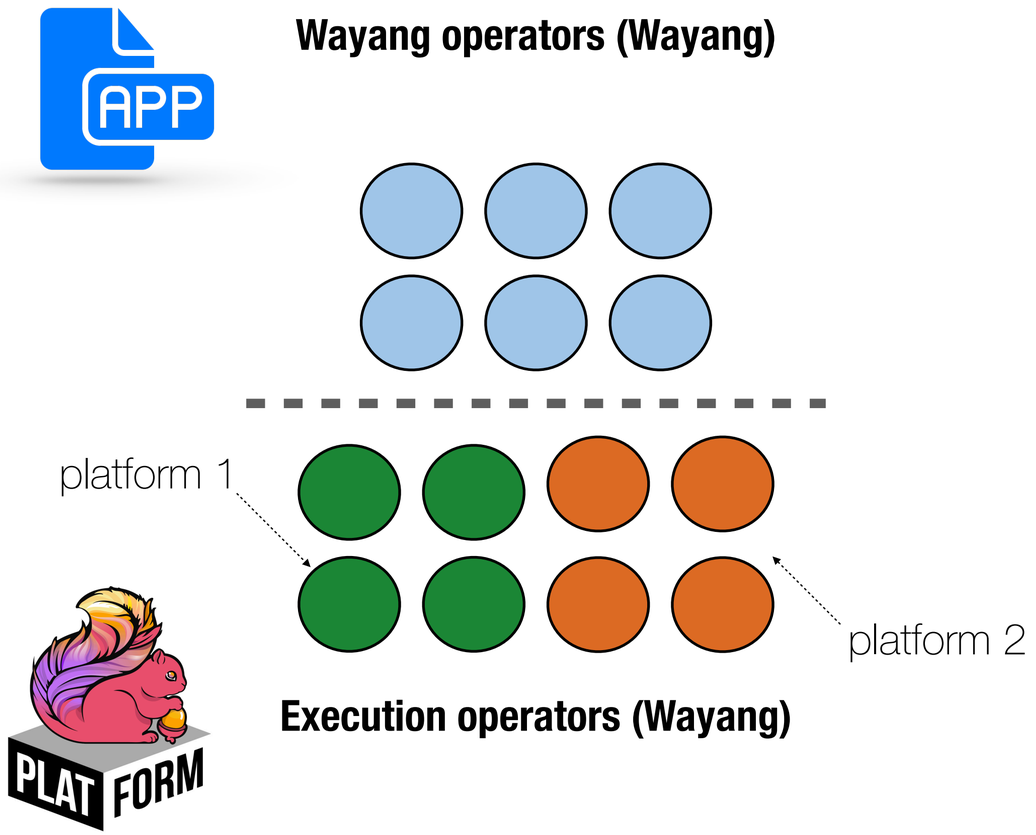
Apache Wayang

2 layers of operators



Apache Wayang

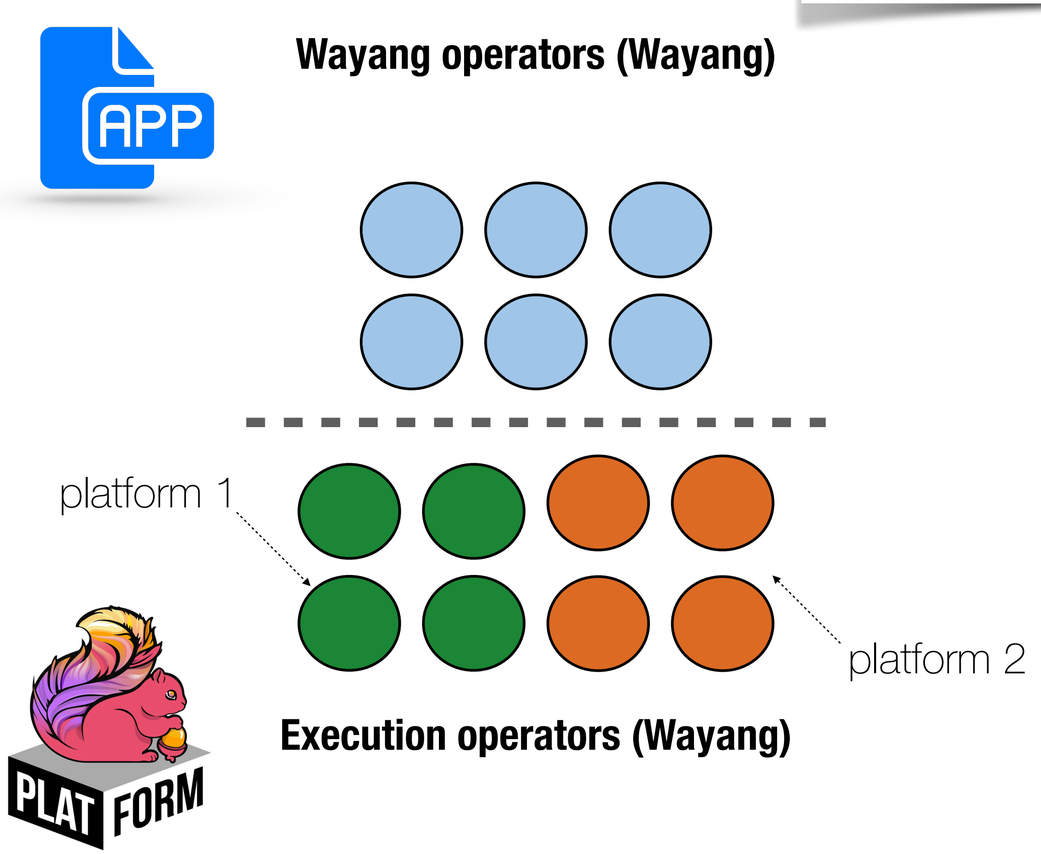
2 layers of operators



Apache Wayang

2 layers of operators

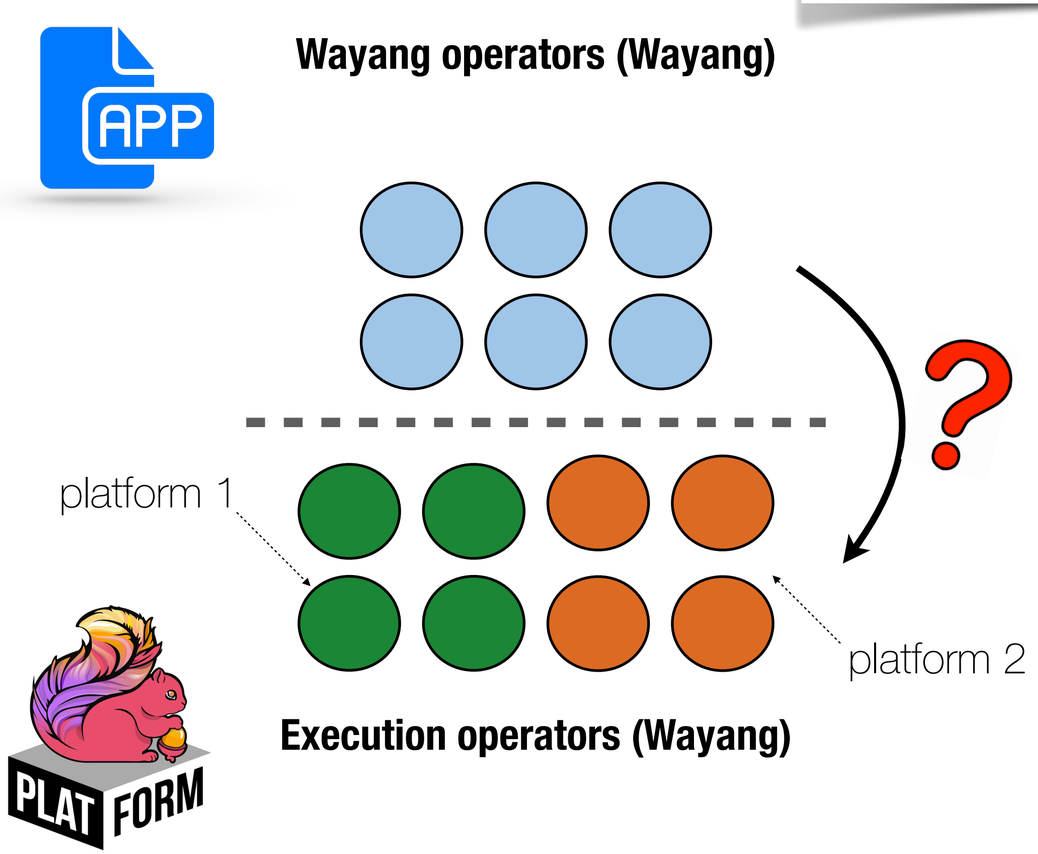
Wayang: fine-granular operators



Apache Wayang

2 layers of operators

Wayang: fine-granular operators

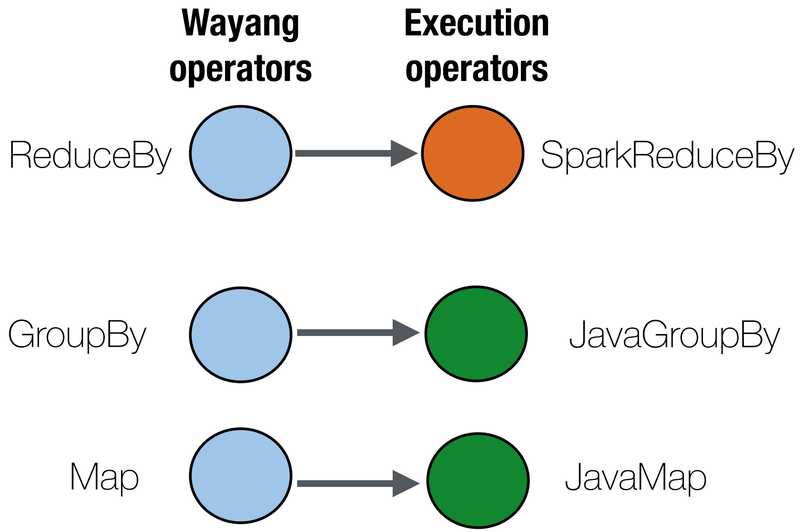


Apache Wayang

Graph-based mappings

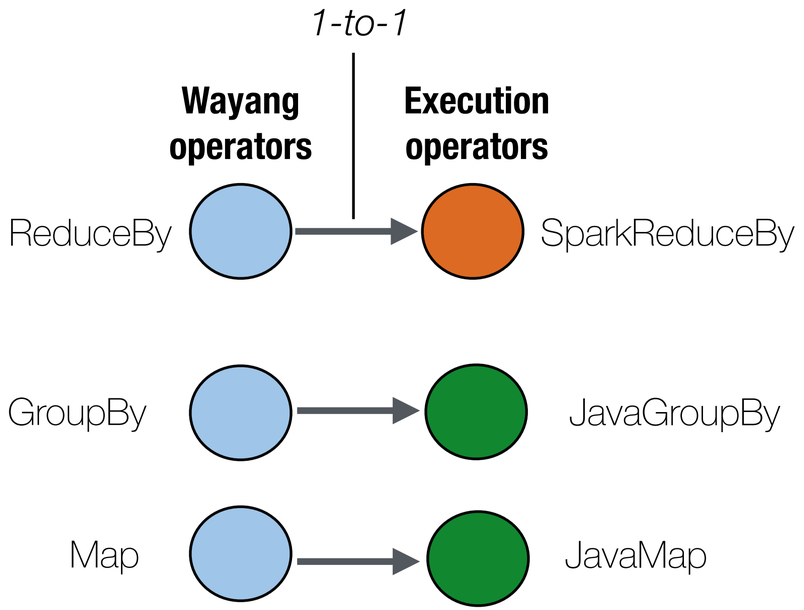
Apache Wayang

Graph-based mappings



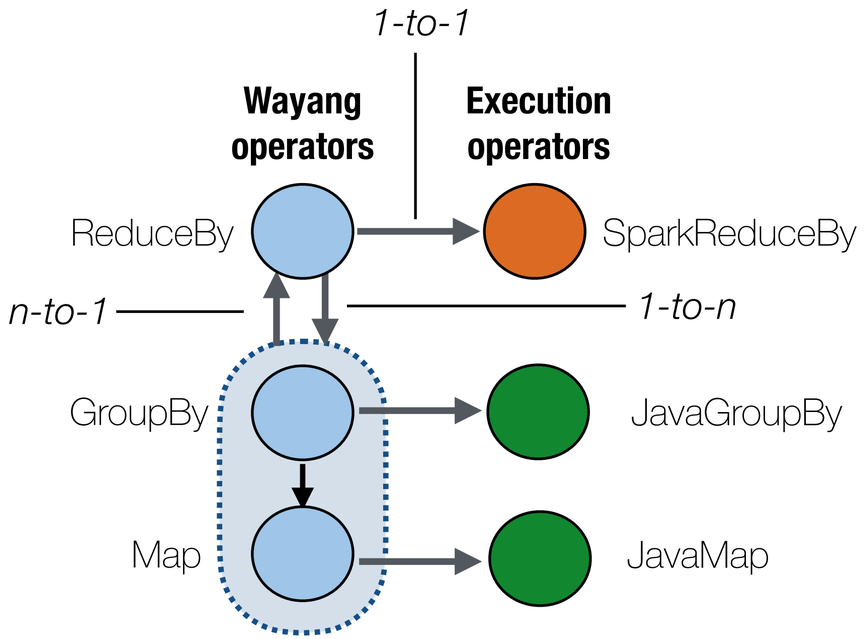
Apache Wayang

Graph-based mappings



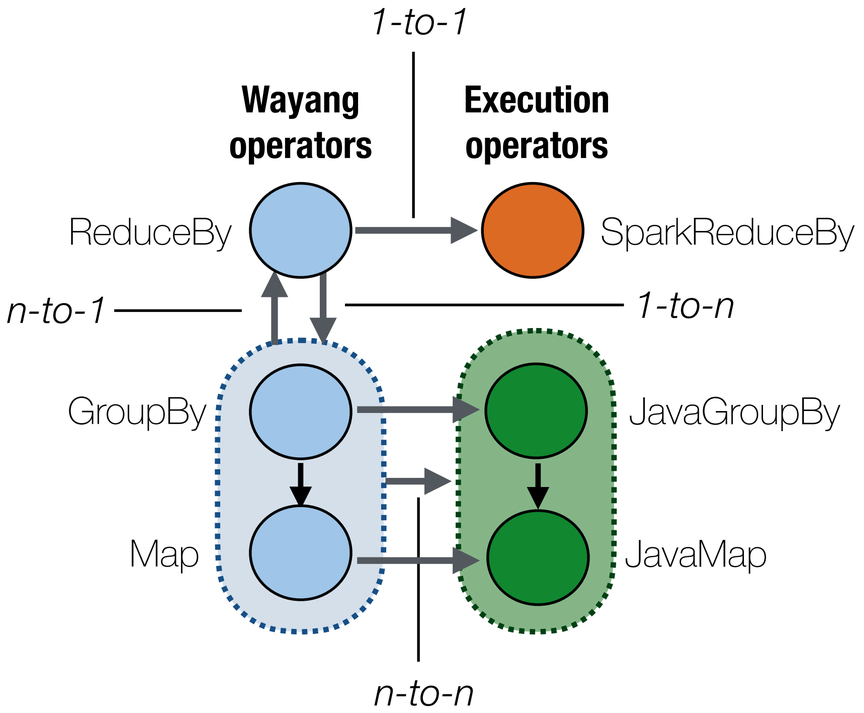
Apache Wayang

Graph-based mappings



Apache Wayang

Graph-based mappings



Challenges

1 Decoupling Applications

4 Extensibility

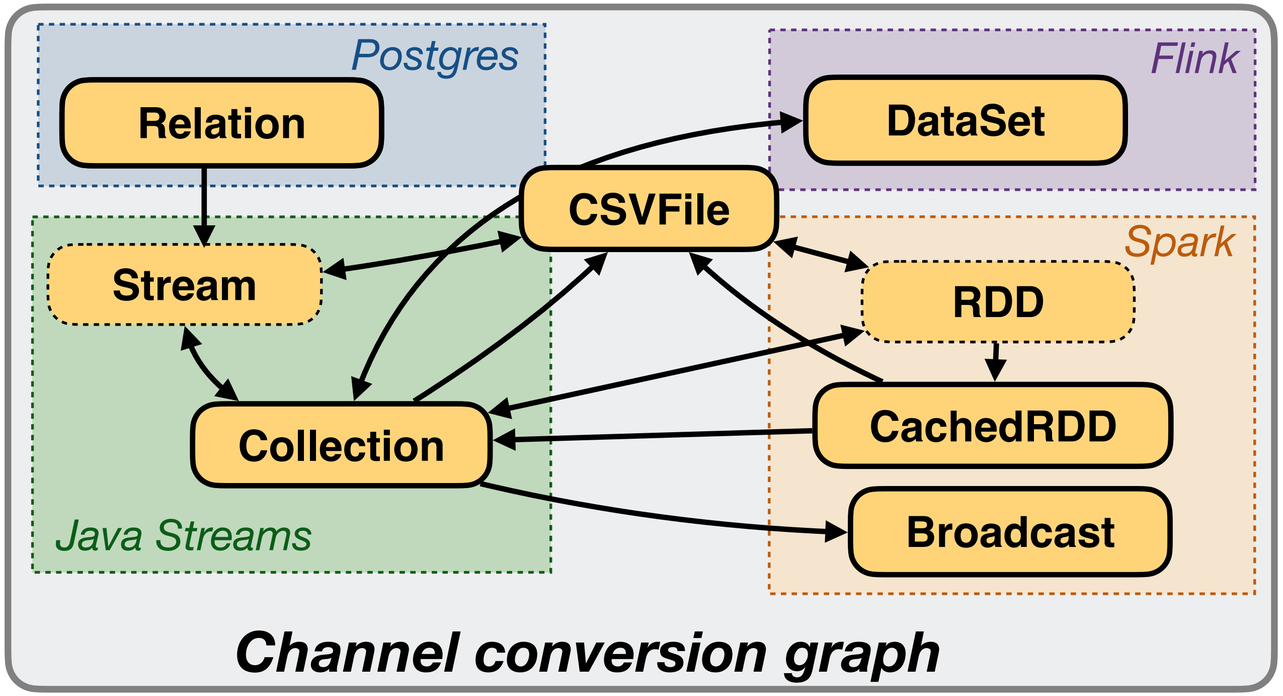
3

Automatic Cross-platform Data Analytics

2 Data Movement

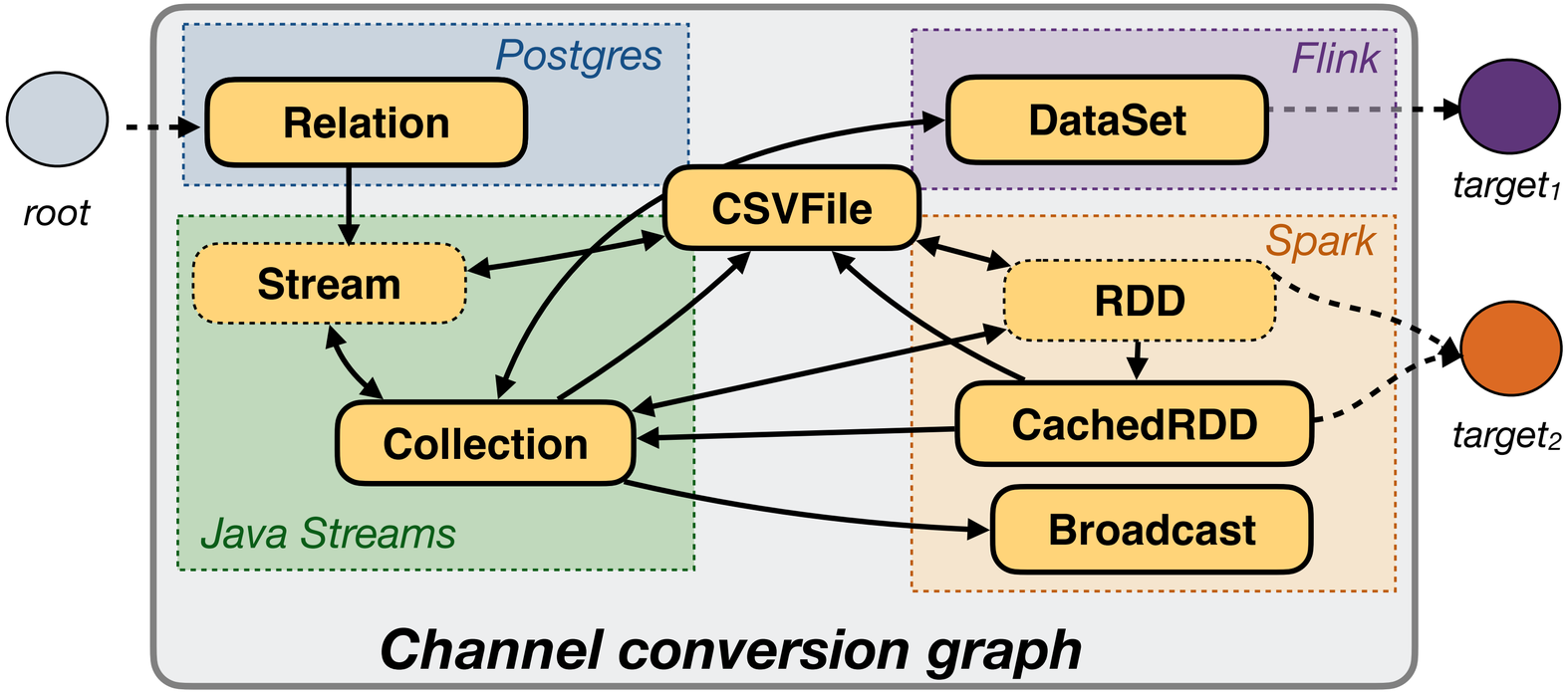
Channels Graph in Apache Wayang

Graph-based data transfer representation



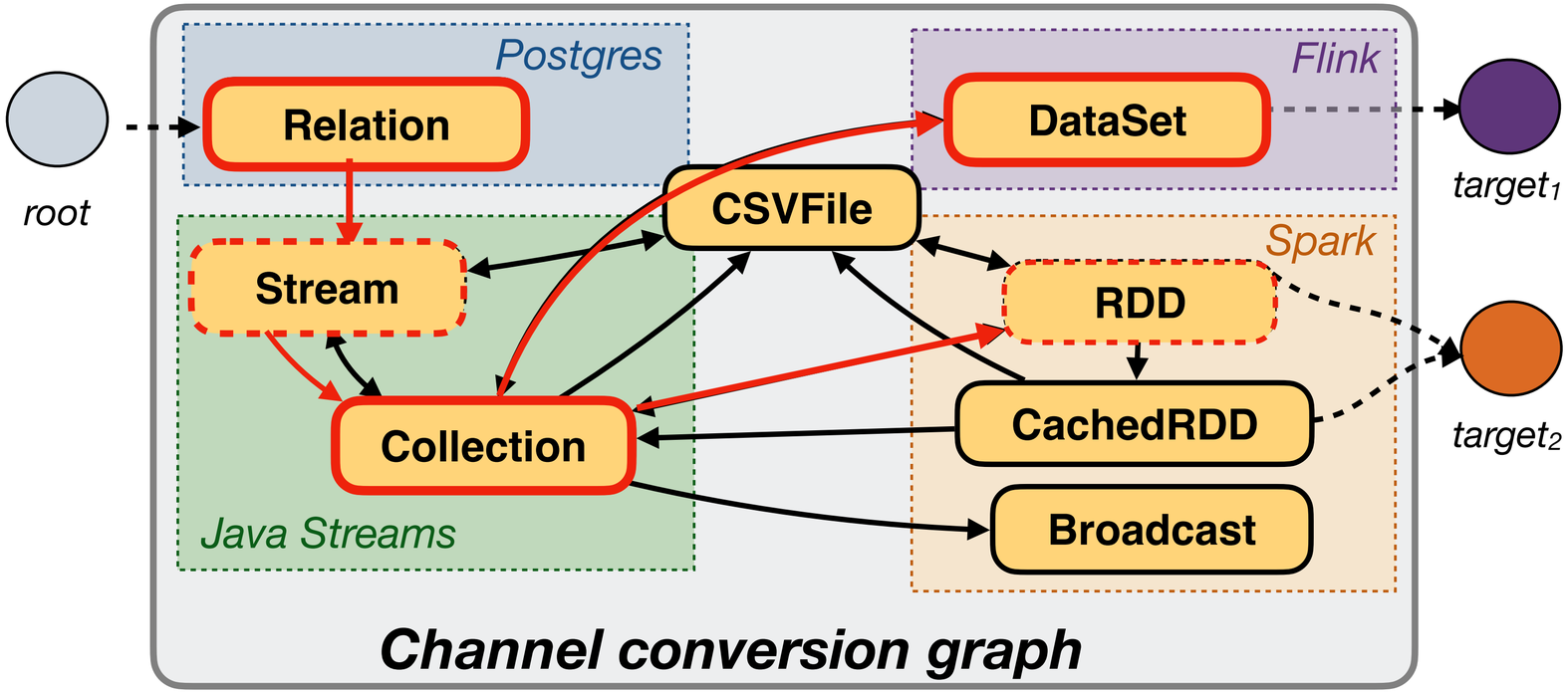
Channels Graph in Apache Wayang

Graph-based data transfer representation



Channels Graph in Apache Wayang

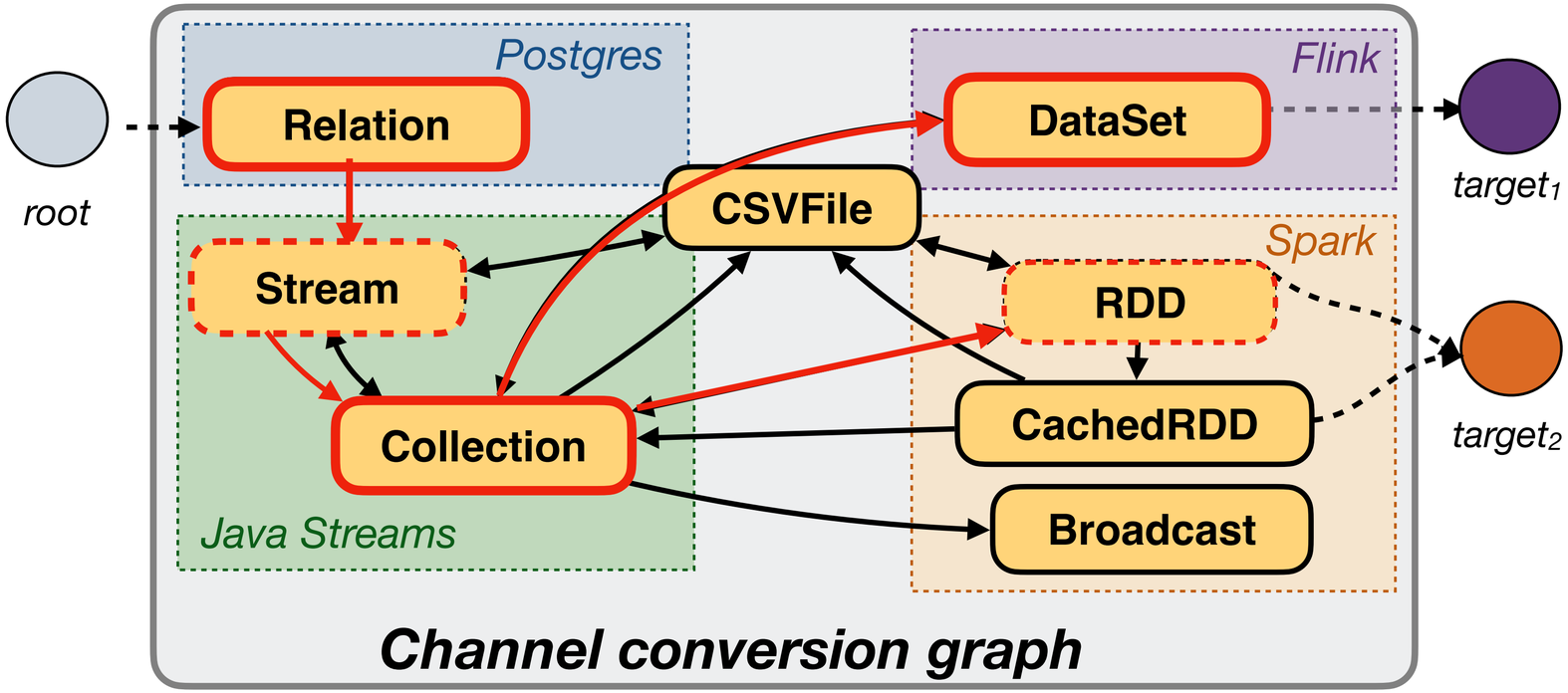
Graph-based data transfer representation



Channels Graph in Apache Wayang

Graph-based data transfer representation

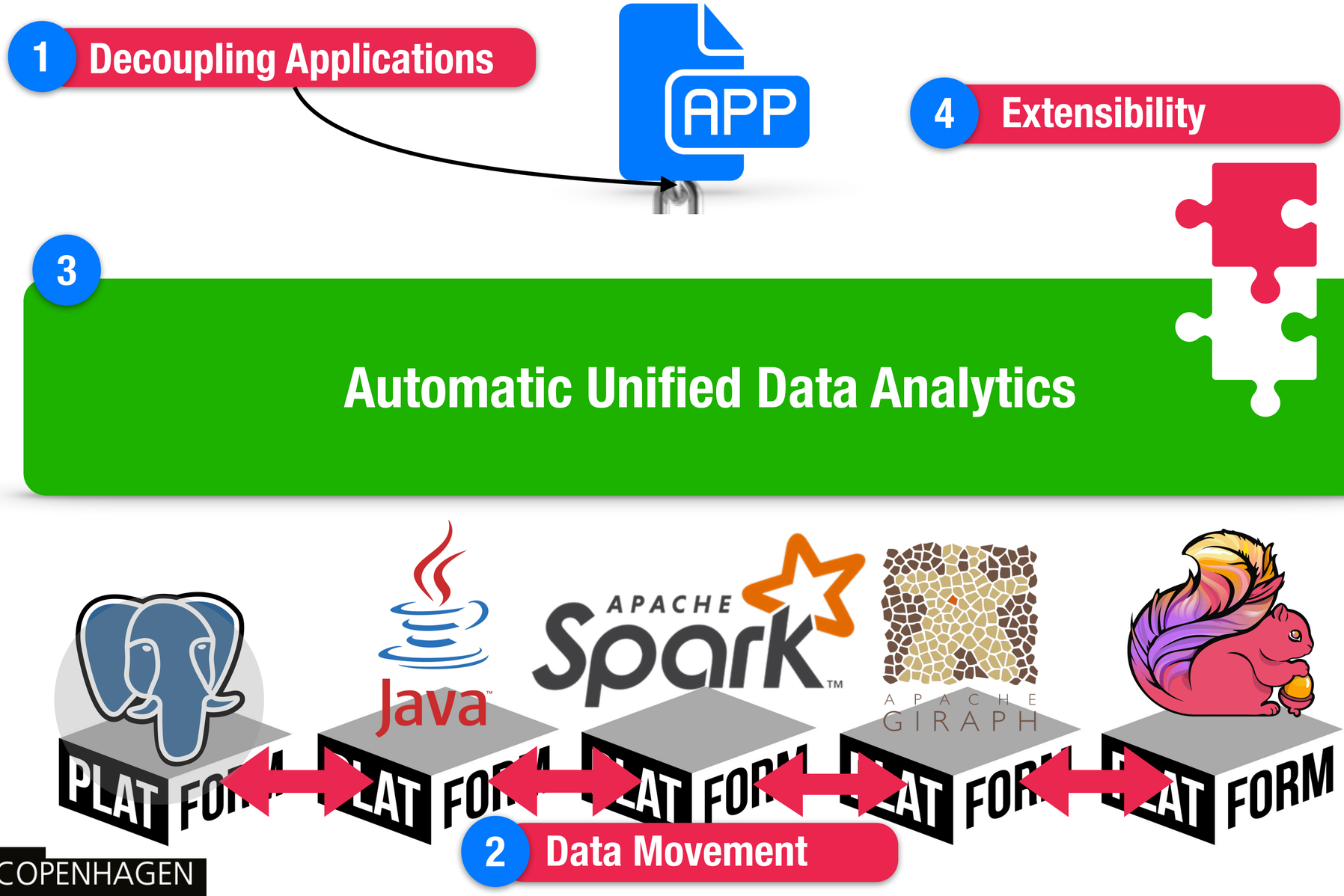
Group Steiner Tree Problem



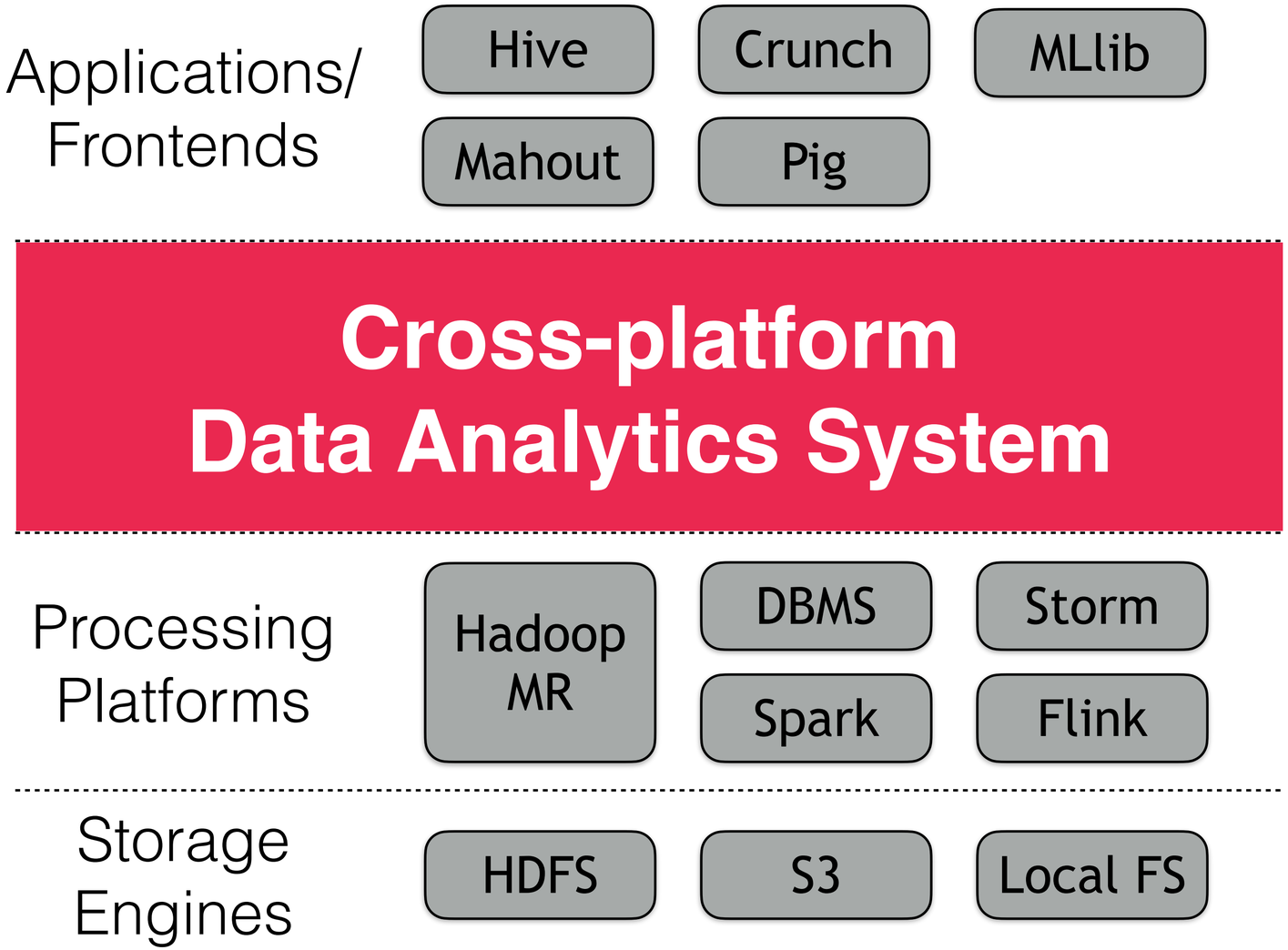
reusable channel

non-reusable channel

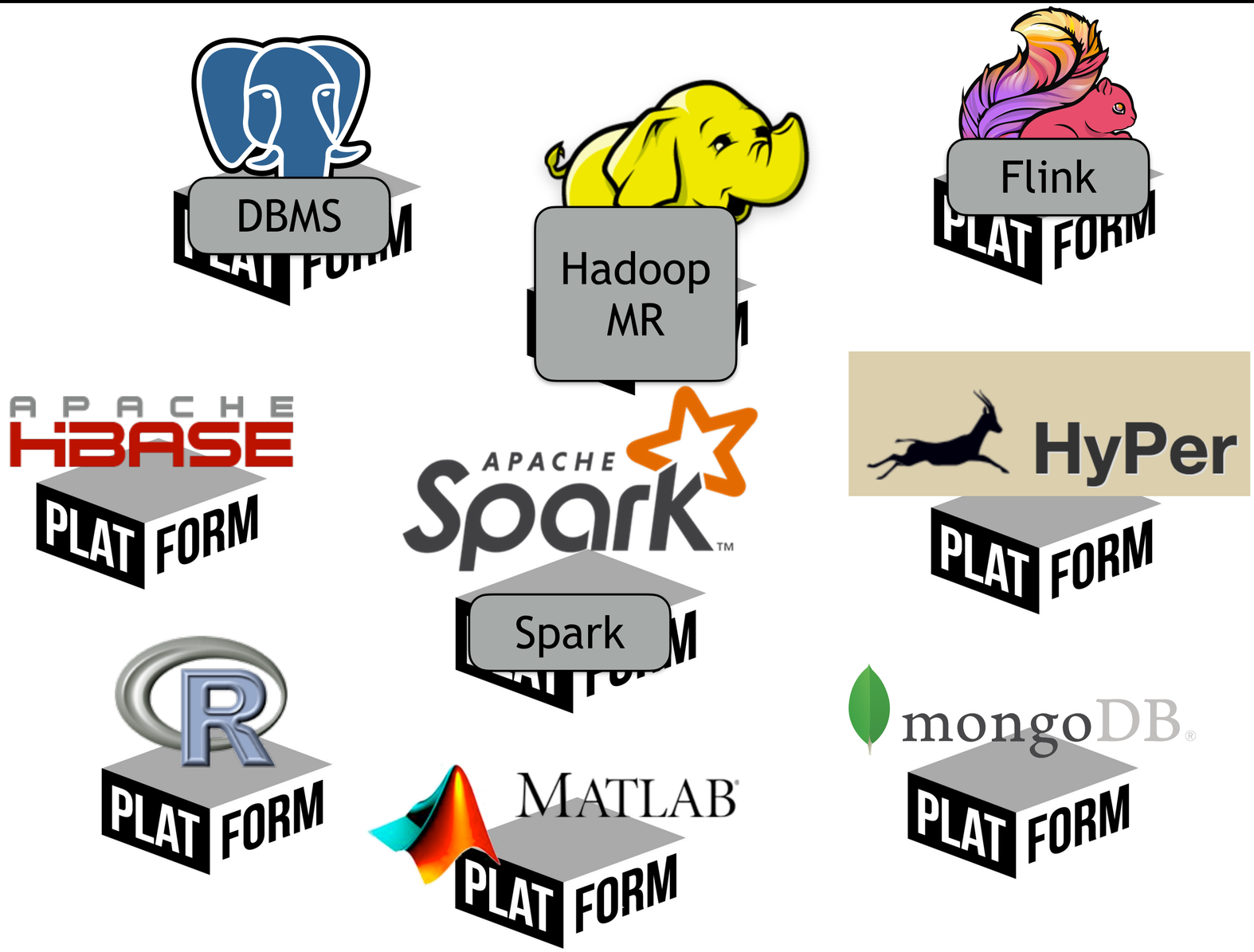
Challenges


1 Decoupling Applications
APP
4 Extensibility
3
Automatic Unified Data Analytics
Java
APACHE Spark
APACHE GIRAPH
PLAT FORM
2 Data Movement

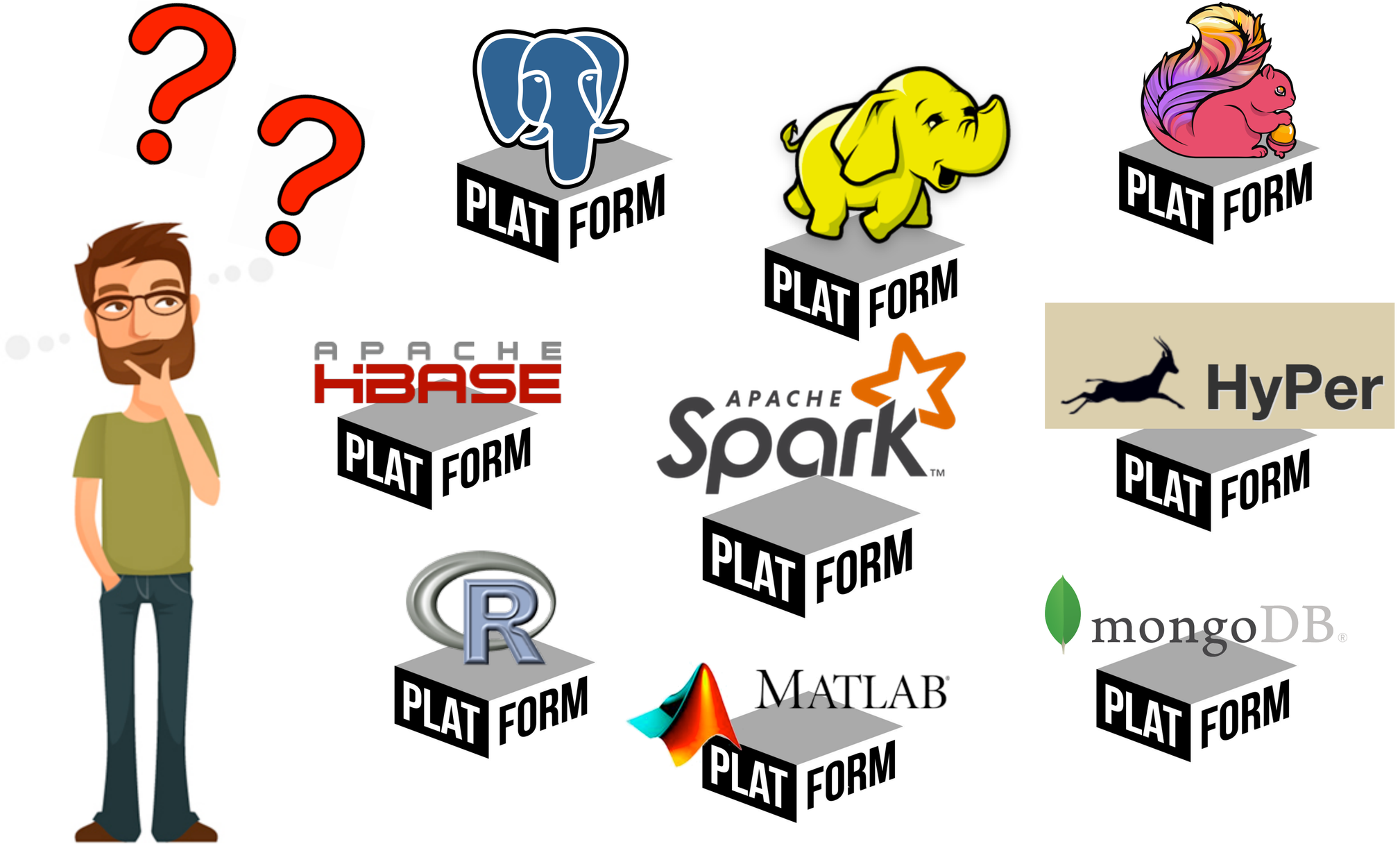
Decoupling Applications



Which Platforms to Use?



Which Platforms to Use?



Unified Data Analytics Query Optimization

Rule-based

Cost-based

Unified Data Analytics Query Optimization

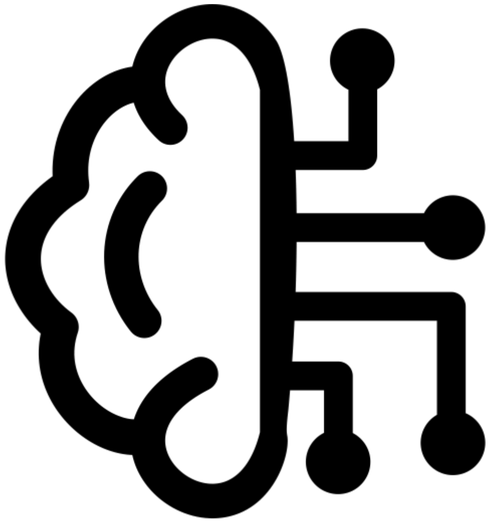
Rule-based

Cost-based

Unified Data Analytics Query Optimization

Rule-based

Cost-based

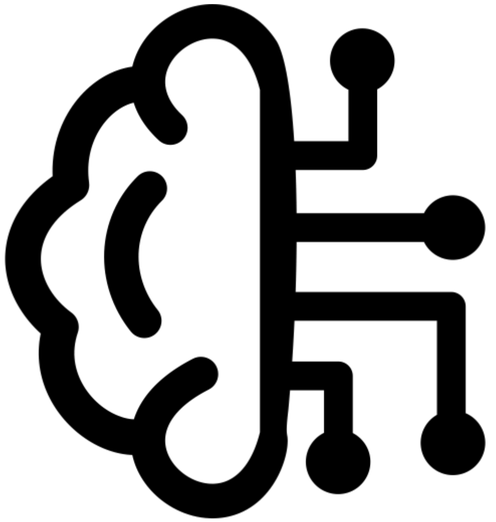


Learning-based

Unified Data Analytics Query Optimization

Rule-based

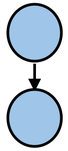
Cost-based



Learning-based

Cost-based QO in Apache Wayang

Wayang plan
(platform-agnostic)



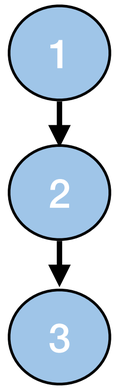
Wayang's optimizer

Cost-based QO in Apache Wayang

Cost-based QO in Apache Wayang

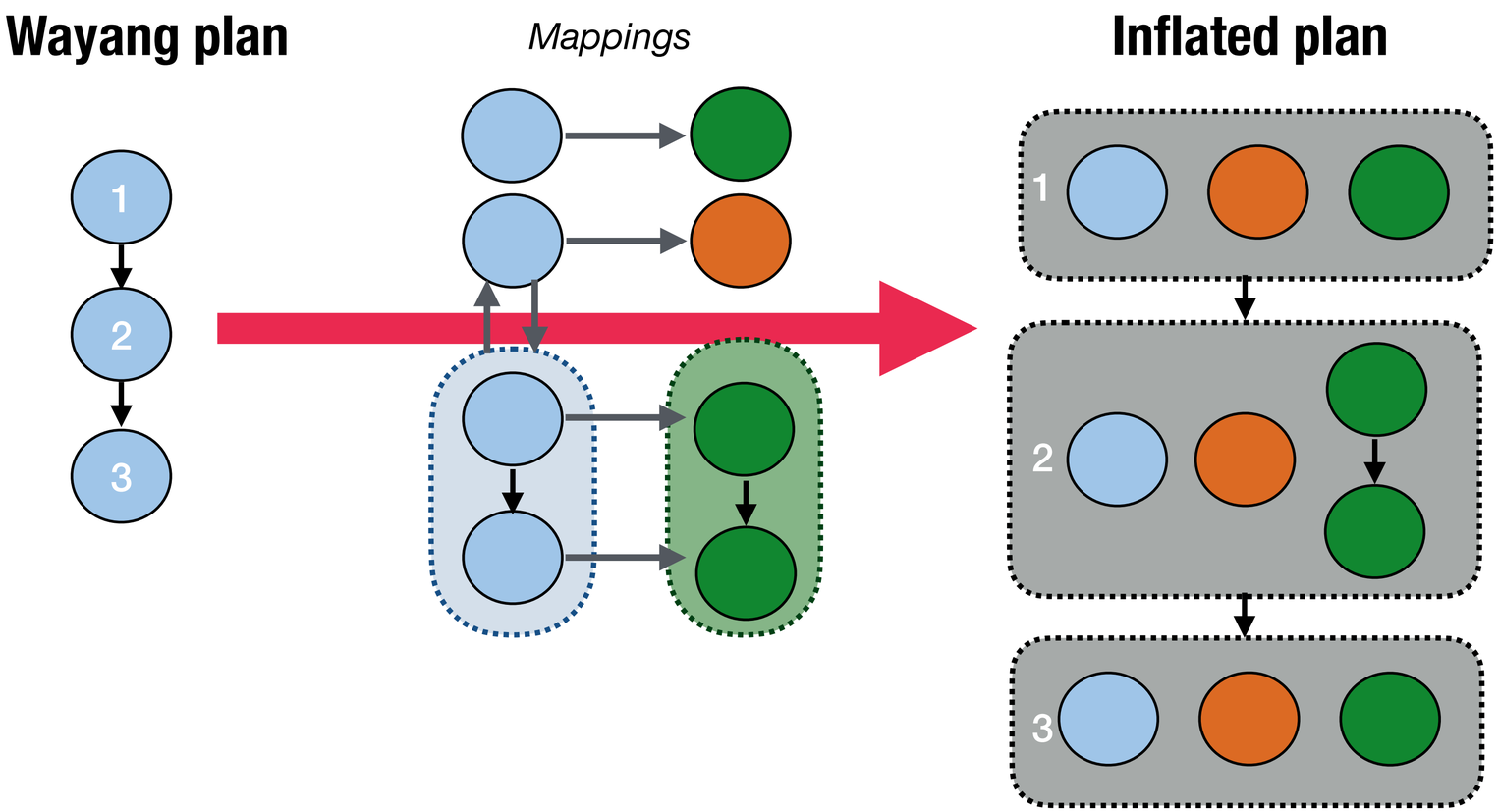
Plan Inflation

Wayang plan



Cost-based QO in Apache Wayang

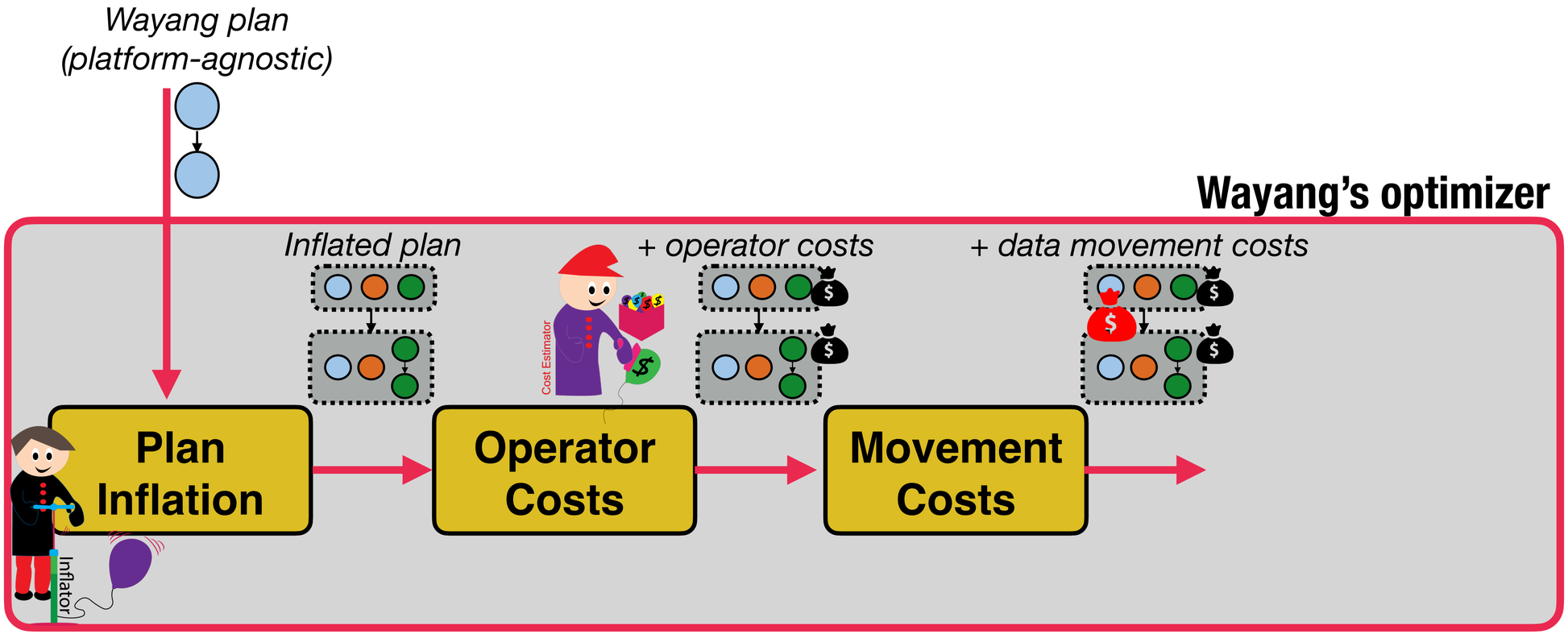
Plan Inflation



Cost-based QO in Apache Wayang

Cost-based QO in Apache Wayang

Cost-based QO in Apache Wayang



Cost-based QO in Apache Wayang

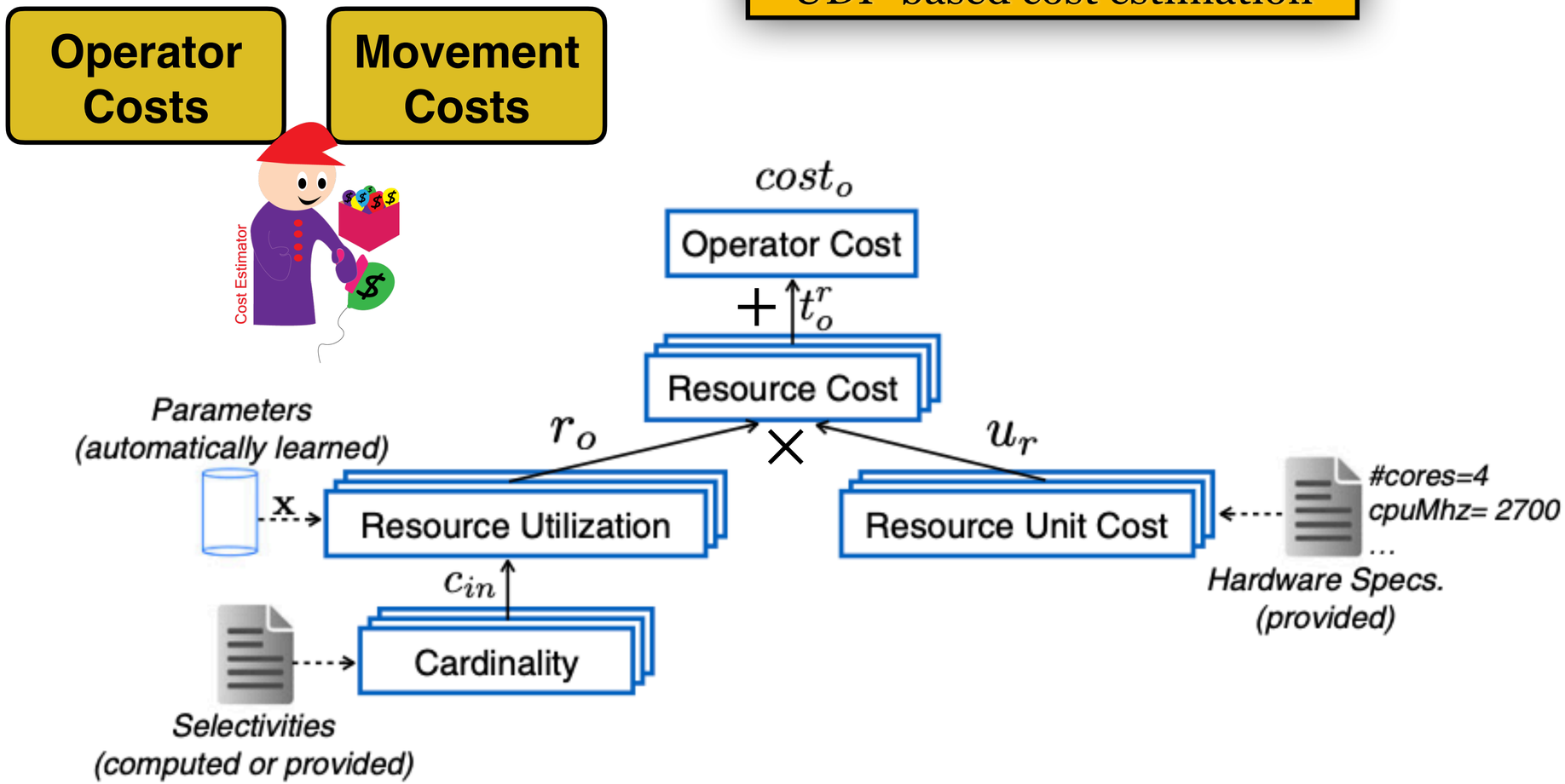
UDF-based cost estimation

Operator Costs

Movement Costs

Cost-based QO in Apache Wayang

UDF-based cost estimation



Cost-based QO in Apache Wayang

UDF-based cost estimation

Operator Costs | **Movement Costs**

Example

$$t^{CPU}_{SparkFilter} = cardIn(Filter) \times \alpha + \beta$$

Cost-based QO in Apache Wayang

UDF-based cost estimation

Operator Costs | **Movement Costs**

Example

$$t^{CPU}_{SparkFilter} = cardIn(Filter) \times \alpha + \beta$$

Learning-based coefficients of cost model

Cost-based QO in Apache Wayang

UDF-based cost estimation

Operator Costs | **Movement Costs**

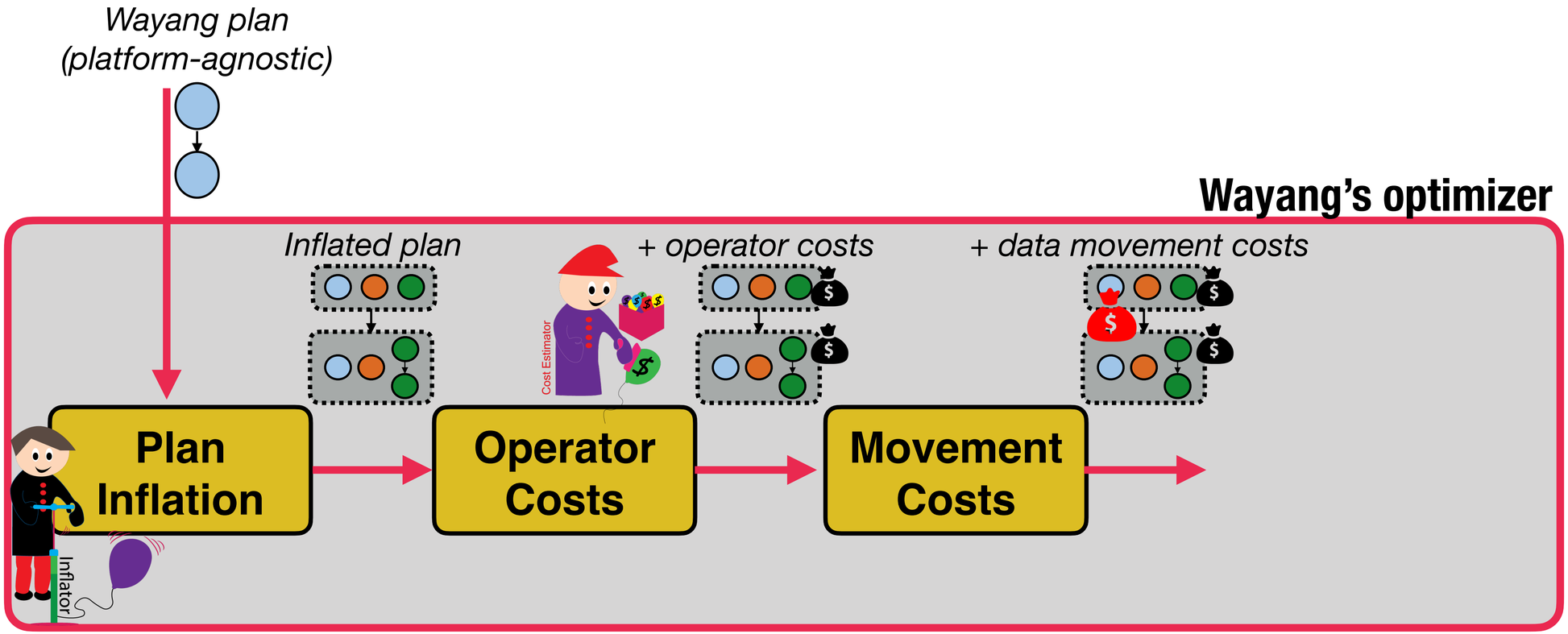
Example

$$t^{CPU}_{SparkFilter} = cardIn(Filter) \times \alpha + \beta$$

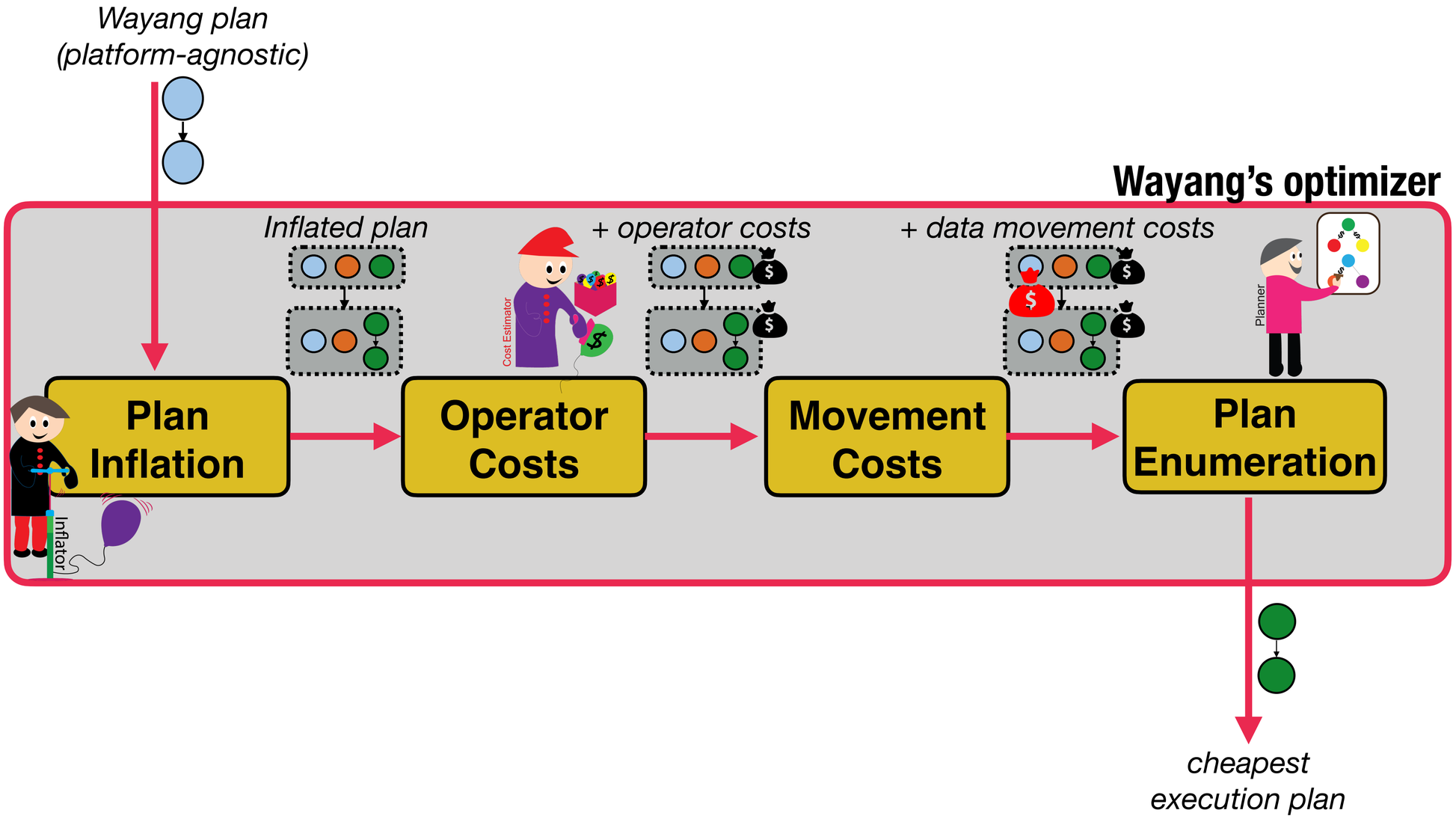
Learning-based coefficients of cost model

Data movement operators are treated as normal operators

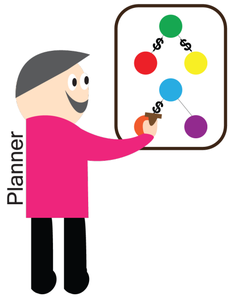
Cost-based QO in Apache Wayang



Cost-based QO in Apache Wayang

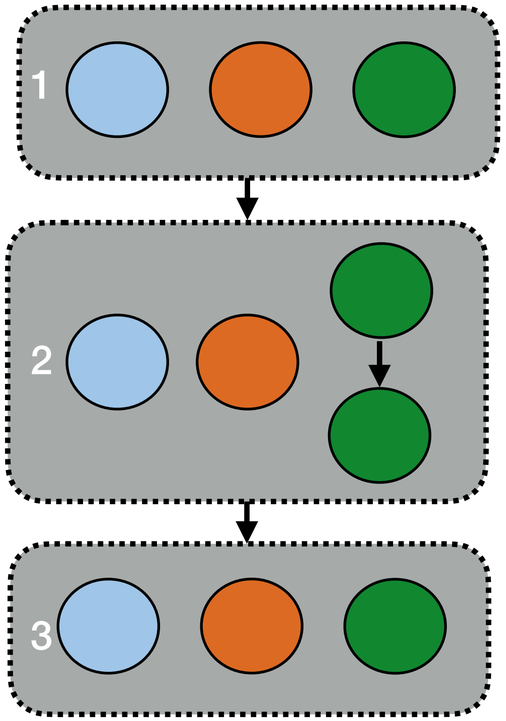


Cost-based QO in Apache Wayang

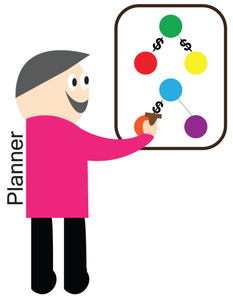


Plan Enumeration

Inflated plan

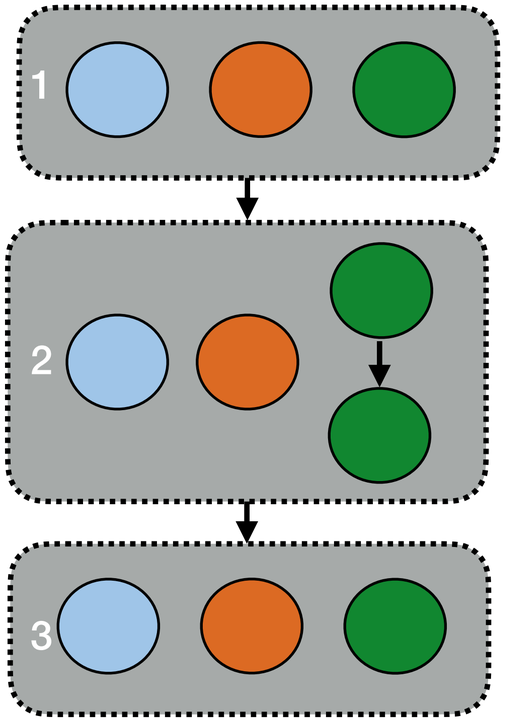


Cost-based QO in Apache Wayang



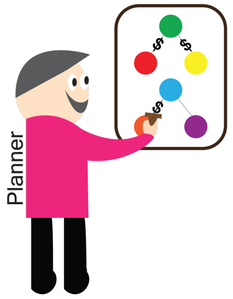
Plan Enumeration

Inflated plan



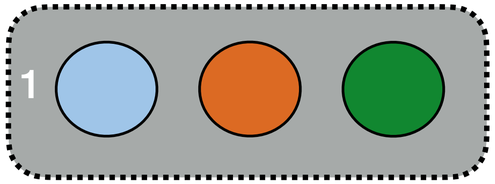
DECOMPOSE

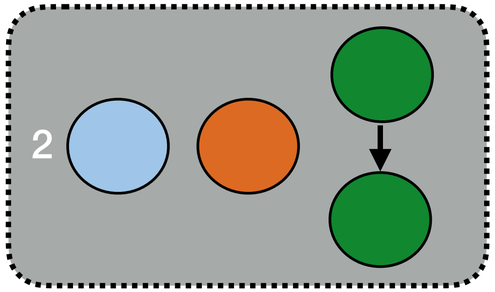
Cost-based QO in Apache Wayang

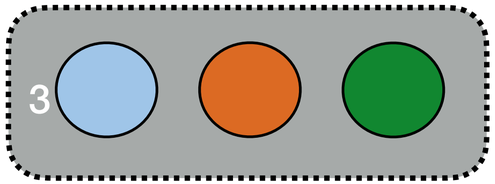


Plan Enumeration

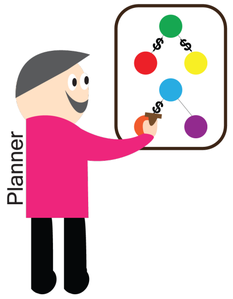
DECOMPOSE



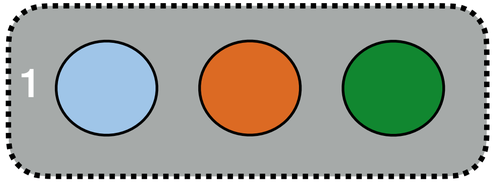


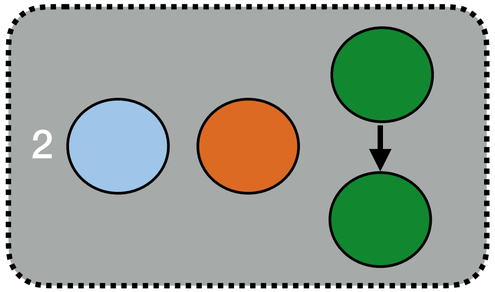


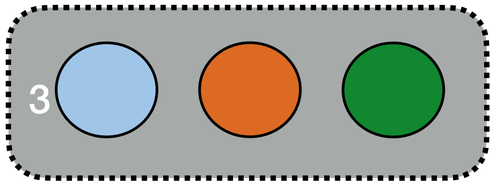
Cost-based QO in Apache Wayang



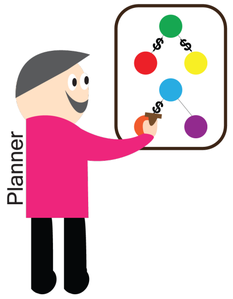
Plan Enumeration





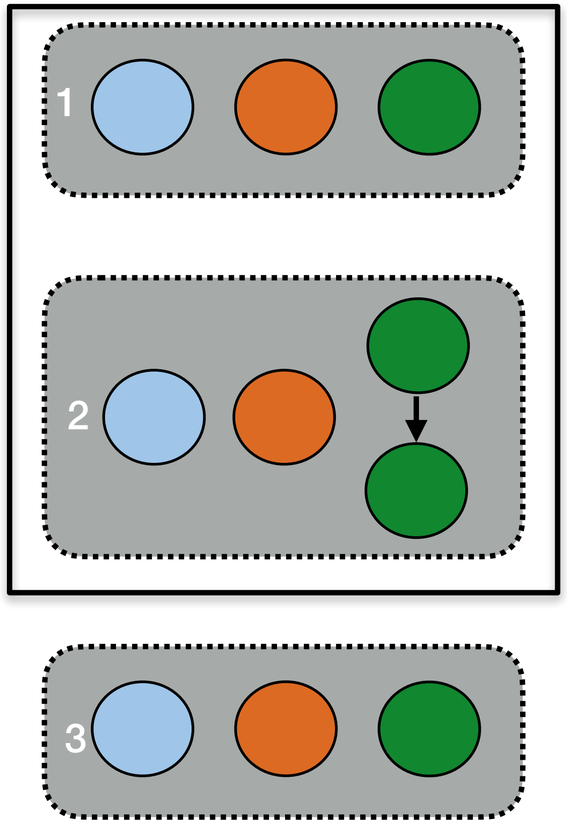


Cost-based QO in Apache Wayang

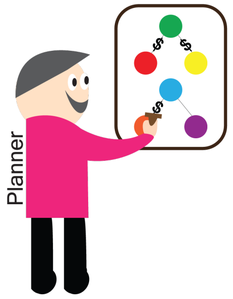


Plan Enumeration

JOIN

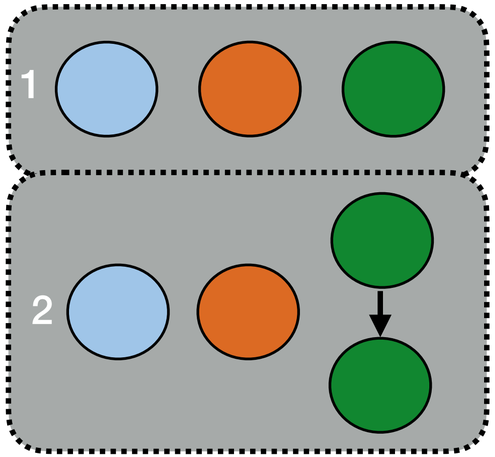


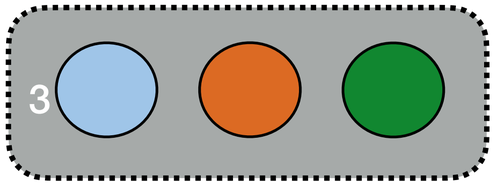
Cost-based QO in Apache Wayang



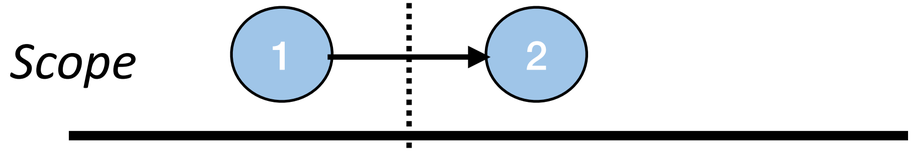
Plan Enumeration

ENUMERATE

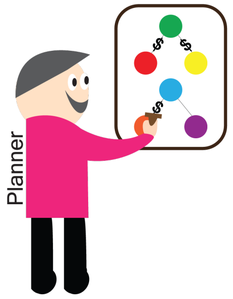




Plan enumeration



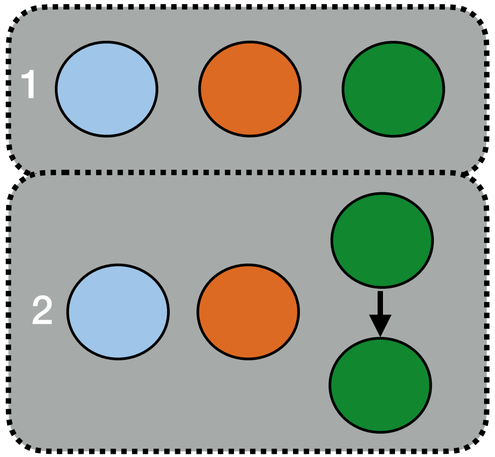
Cost-based QO in Apache Wayang

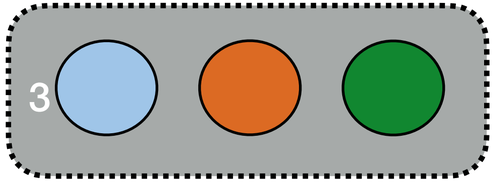


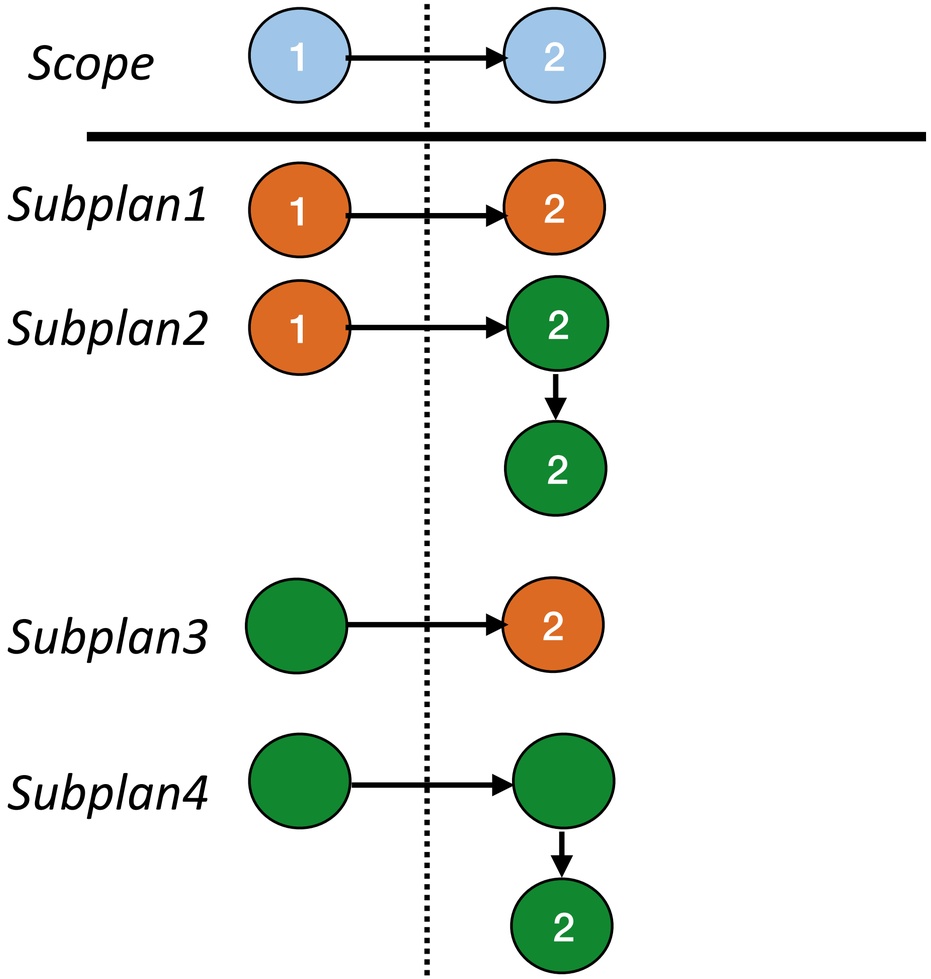
Plan Enumeration

ENUMERATE

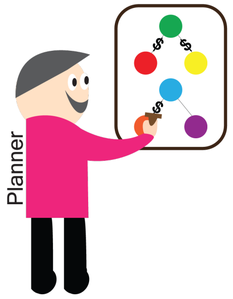
Plan enumeration





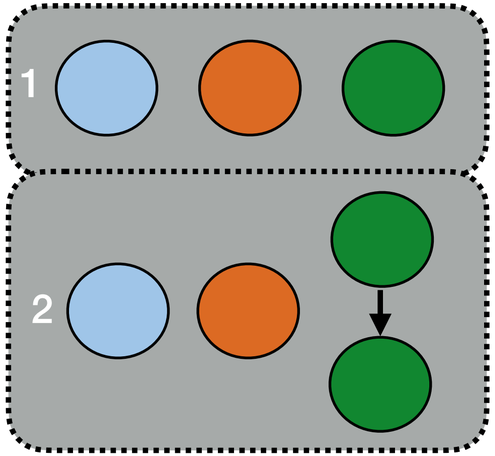


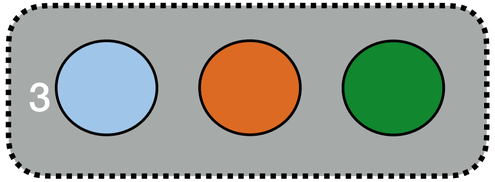
Cost-based QO in Apache Wayang



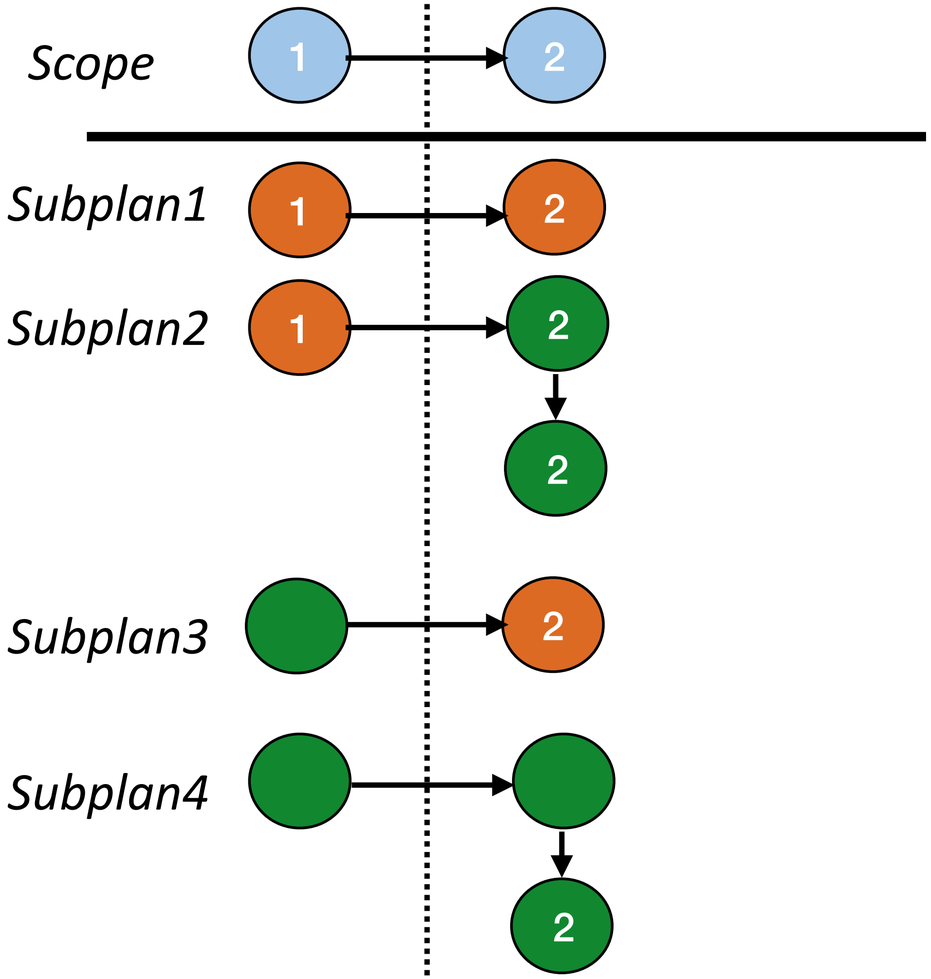
Plan Enumeration

PRUNE

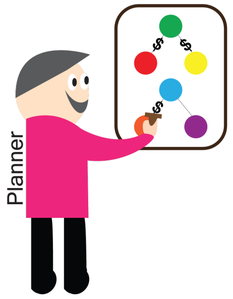




Plan enumeration

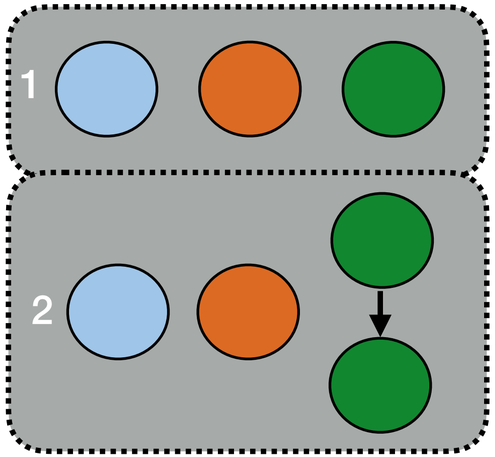


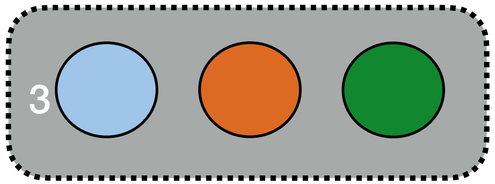
Cost-based QO in Apache Wayang



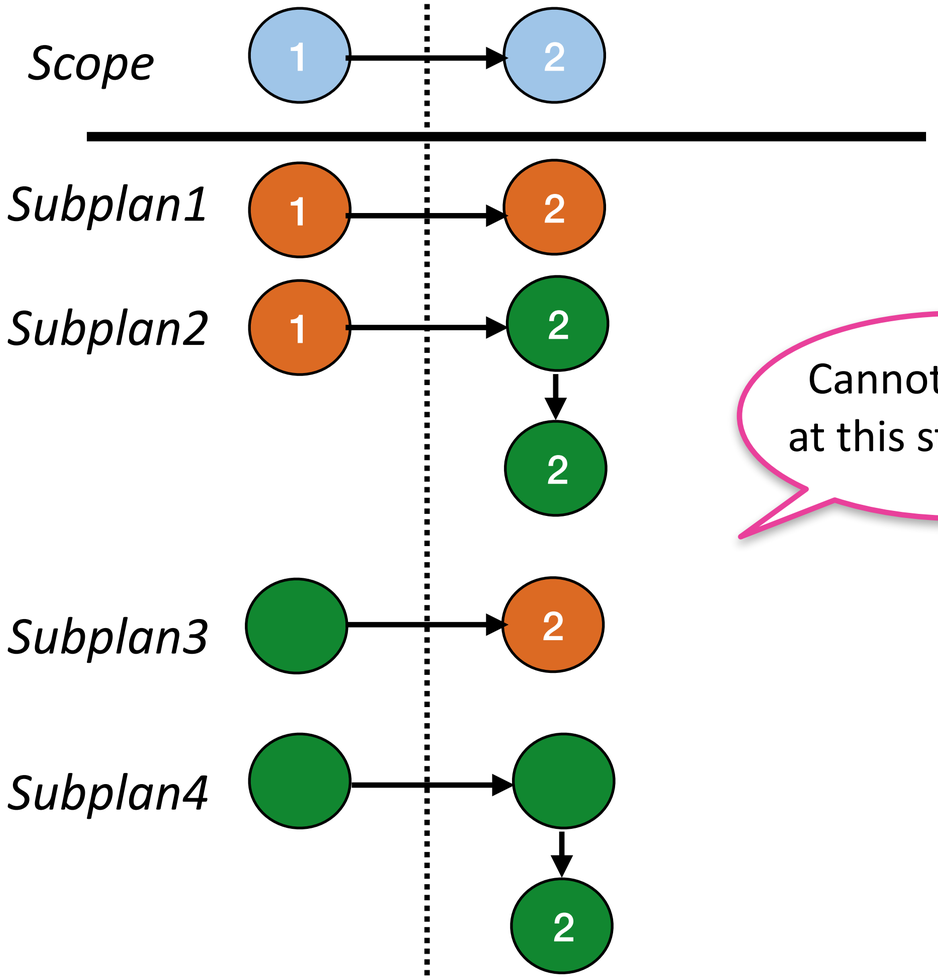
Plan Enumeration

PRUNE



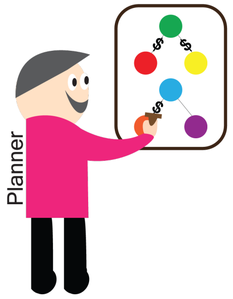


Plan enumeration



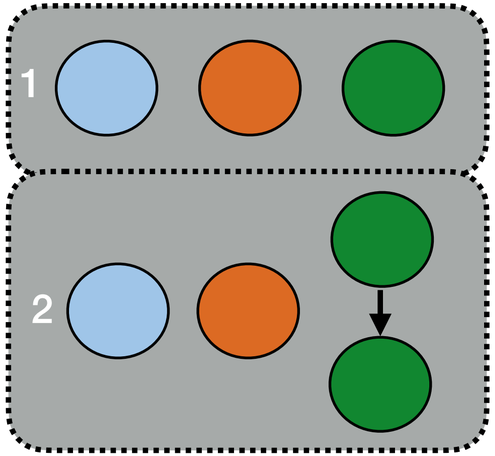
Cannot prune at this stage

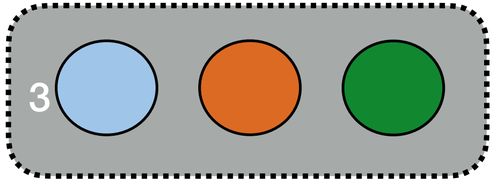
Cost-based QO in Apache Wayang



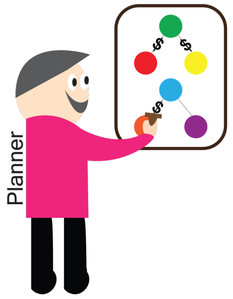
Plan Enumeration

JOIN



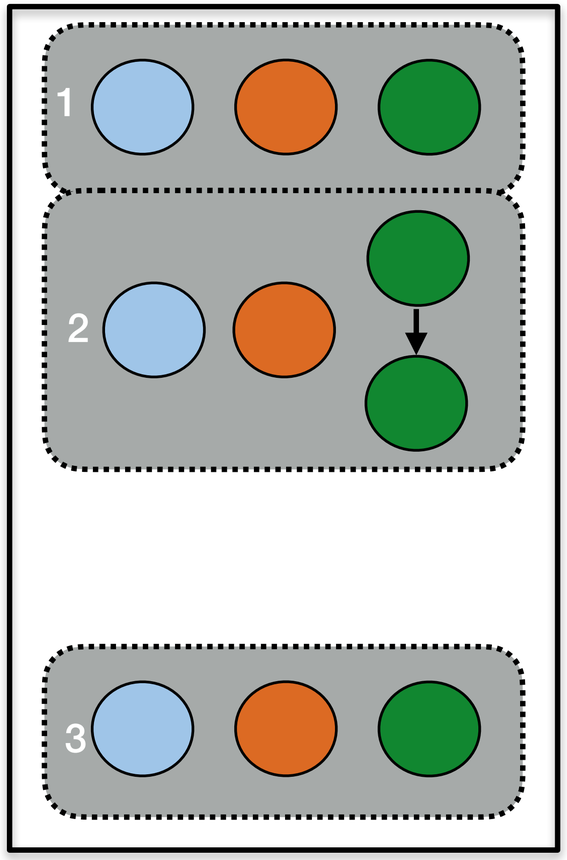


Cost-based QO in Apache Wayang

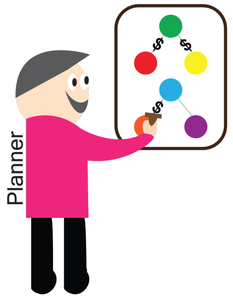


Plan Enumeration

JOIN

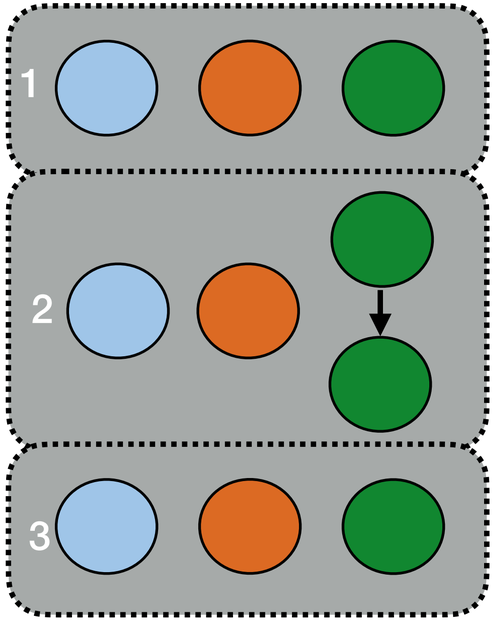


Cost-based QO in Apache Wayang

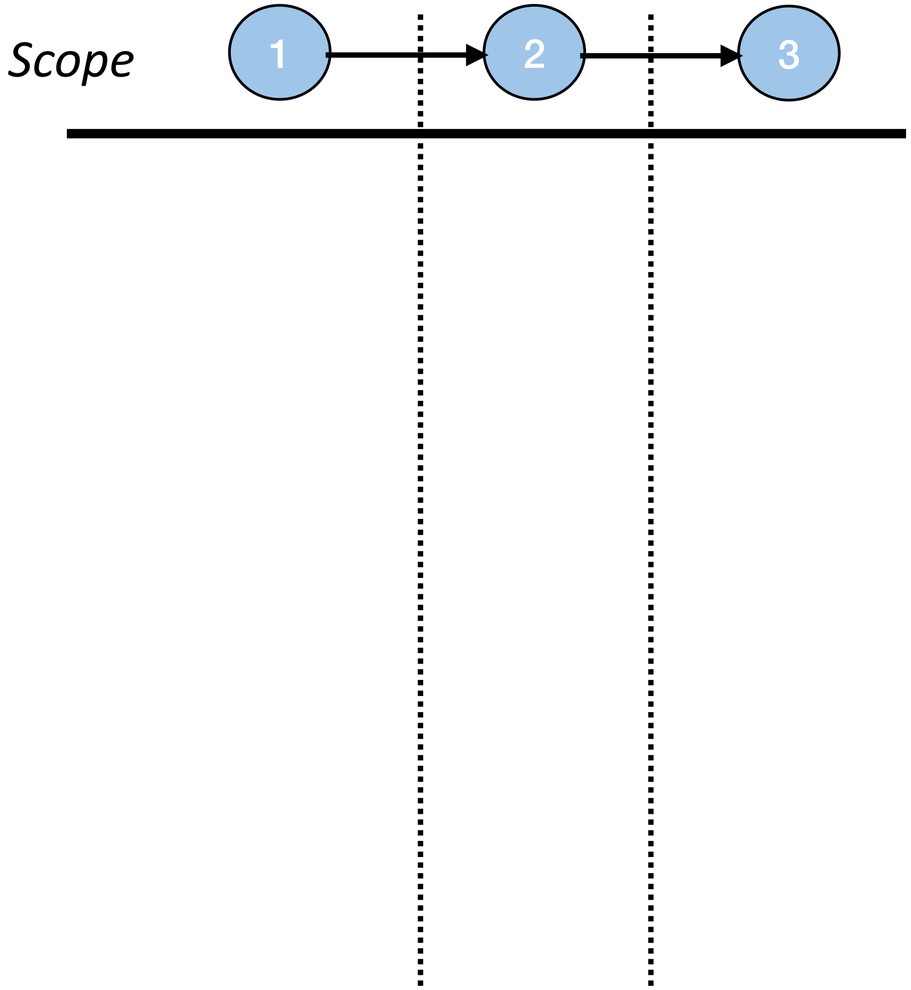


Plan Enumeration

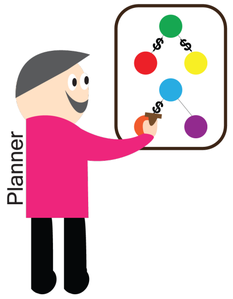
ENUMERATE



Plan enumeration

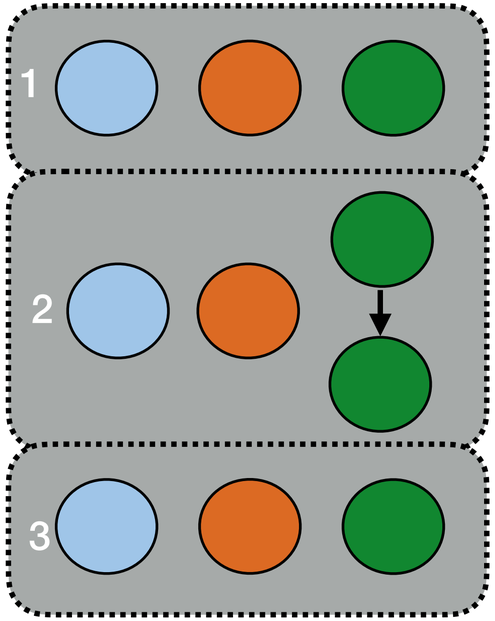


Cost-based QO in Apache Wayang

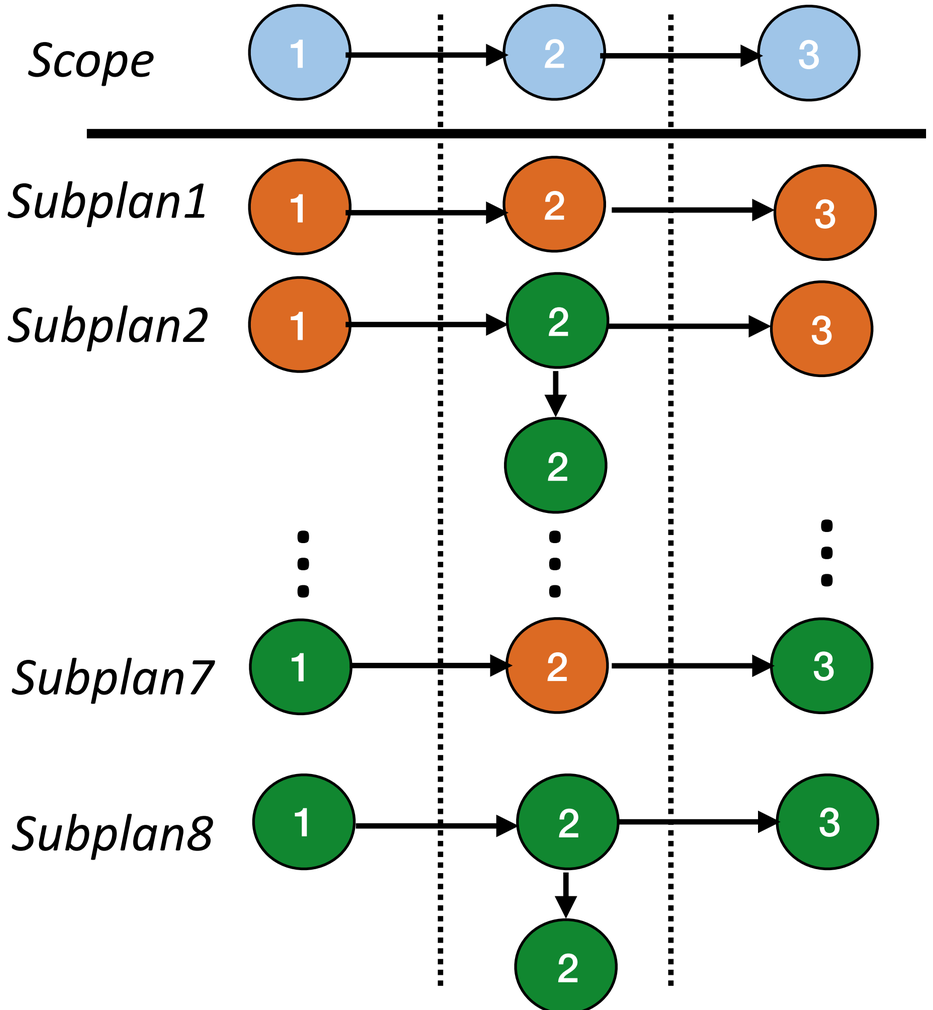


Plan Enumeration

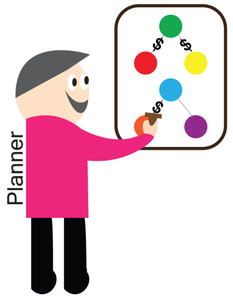
ENUMERATE



Plan enumeration

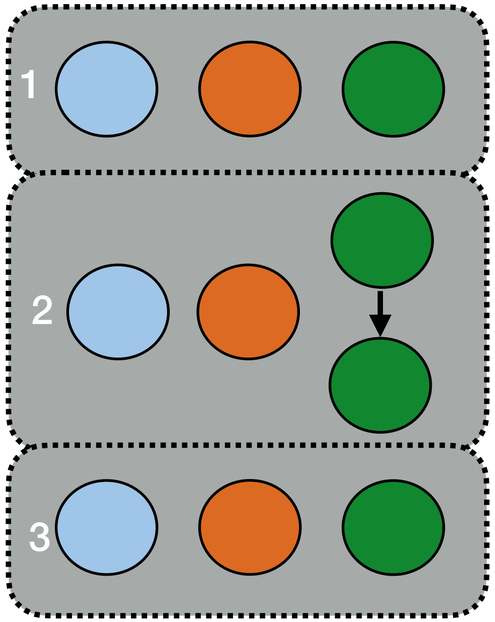


Cost-based QO in Apache Wayang

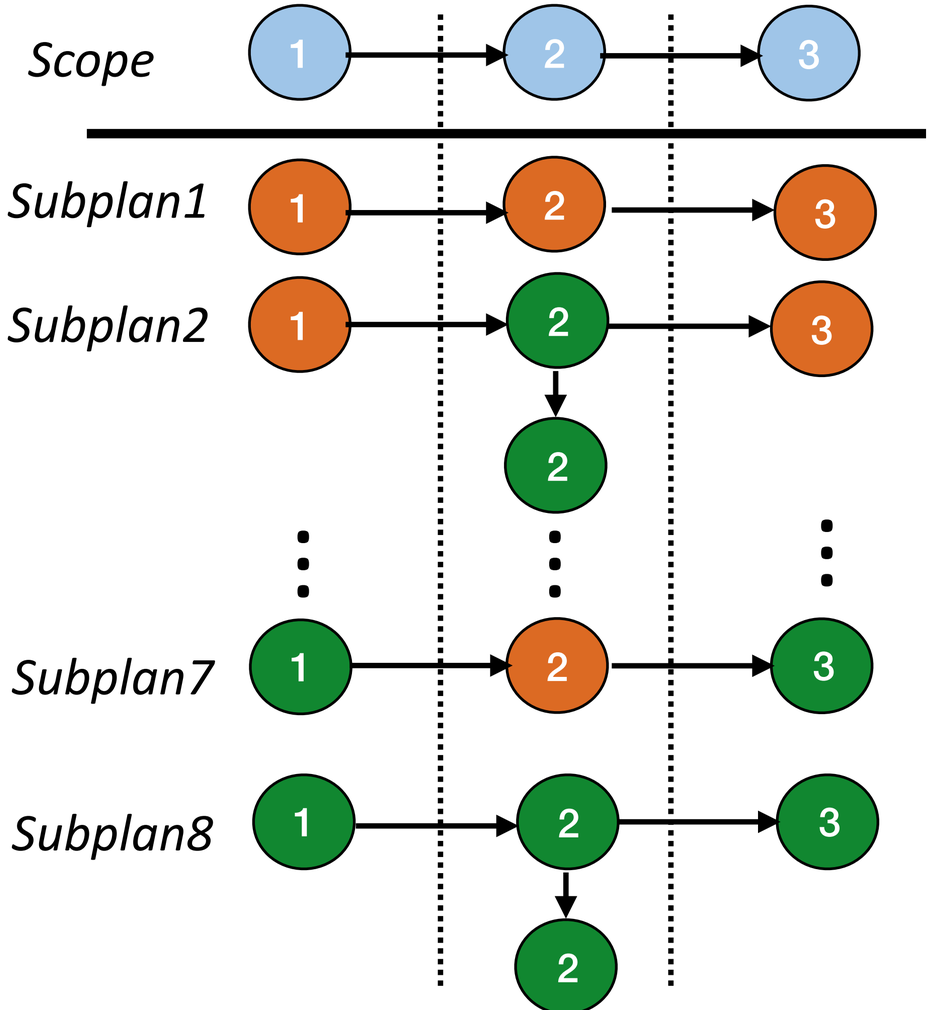


Plan Enumeration

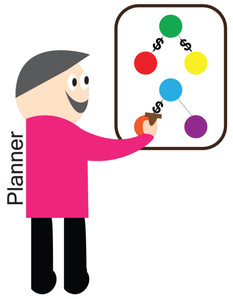
PRUNE



Plan enumeration

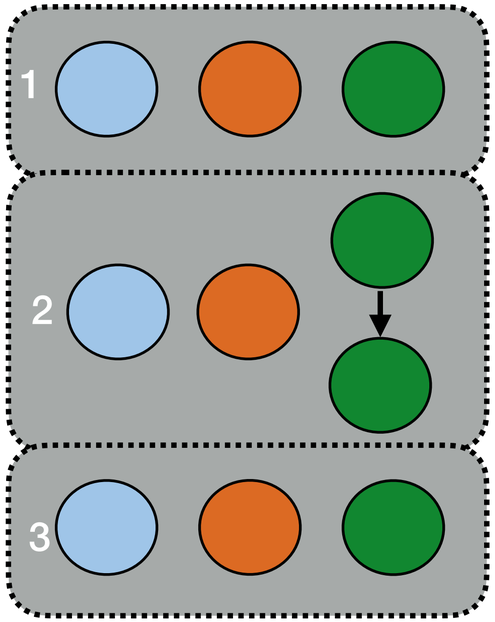


Cost-based QO in Apache Wayang

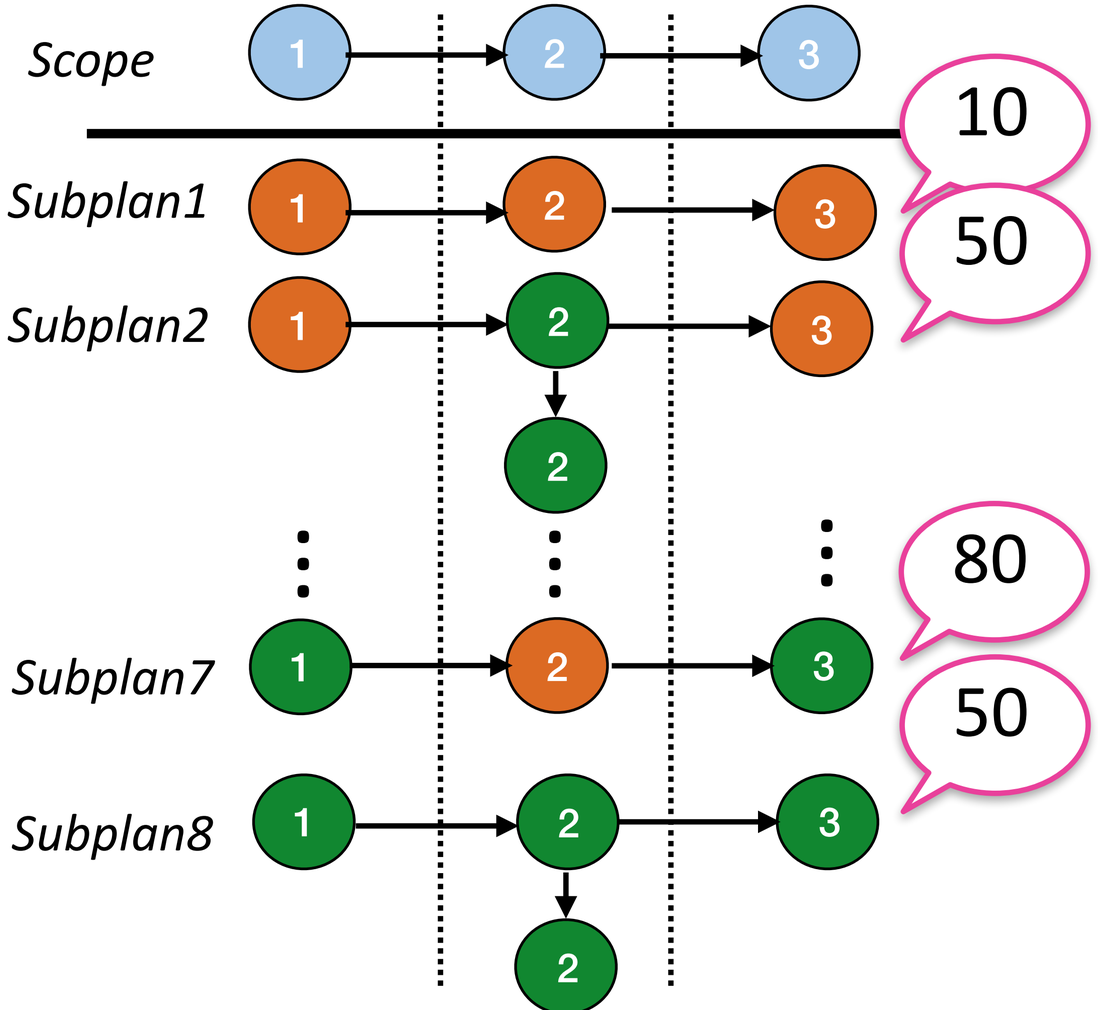


Plan Enumeration

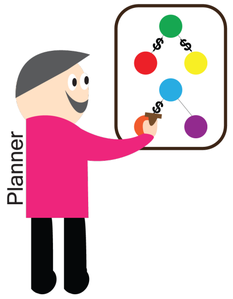
PRUNE



Plan enumeration

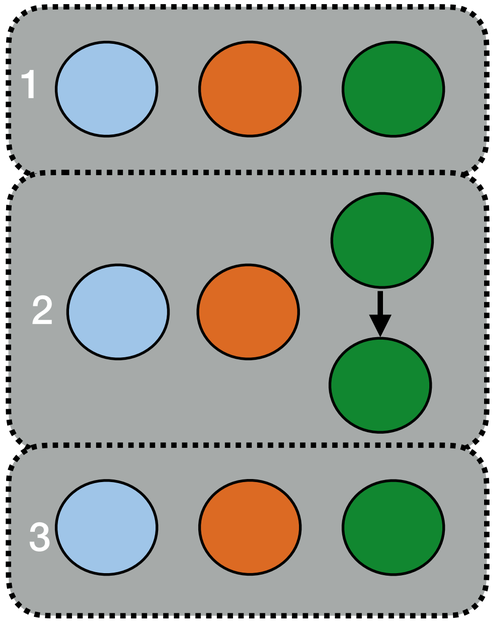


Cost-based QO in Apache Wayang

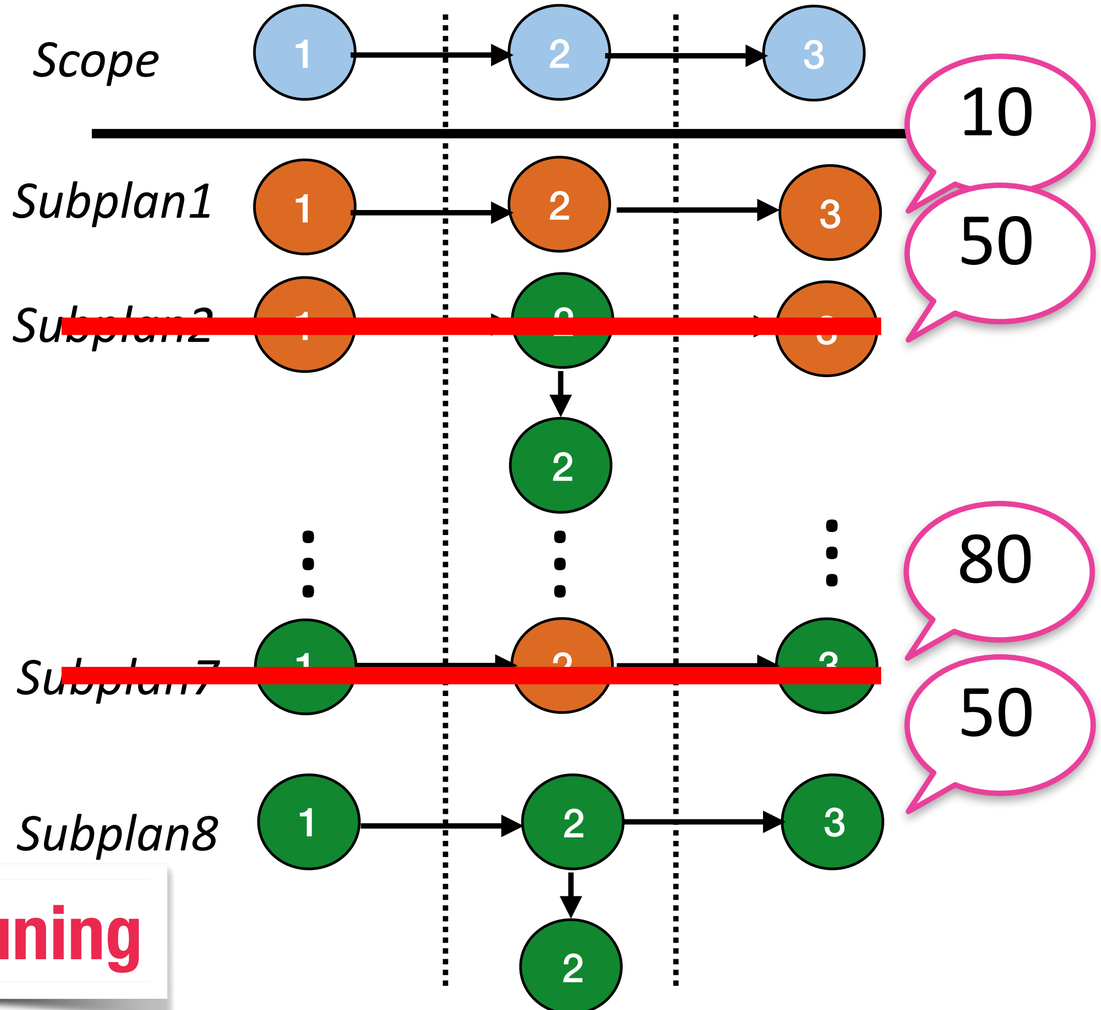


Plan Enumeration

PRUNE

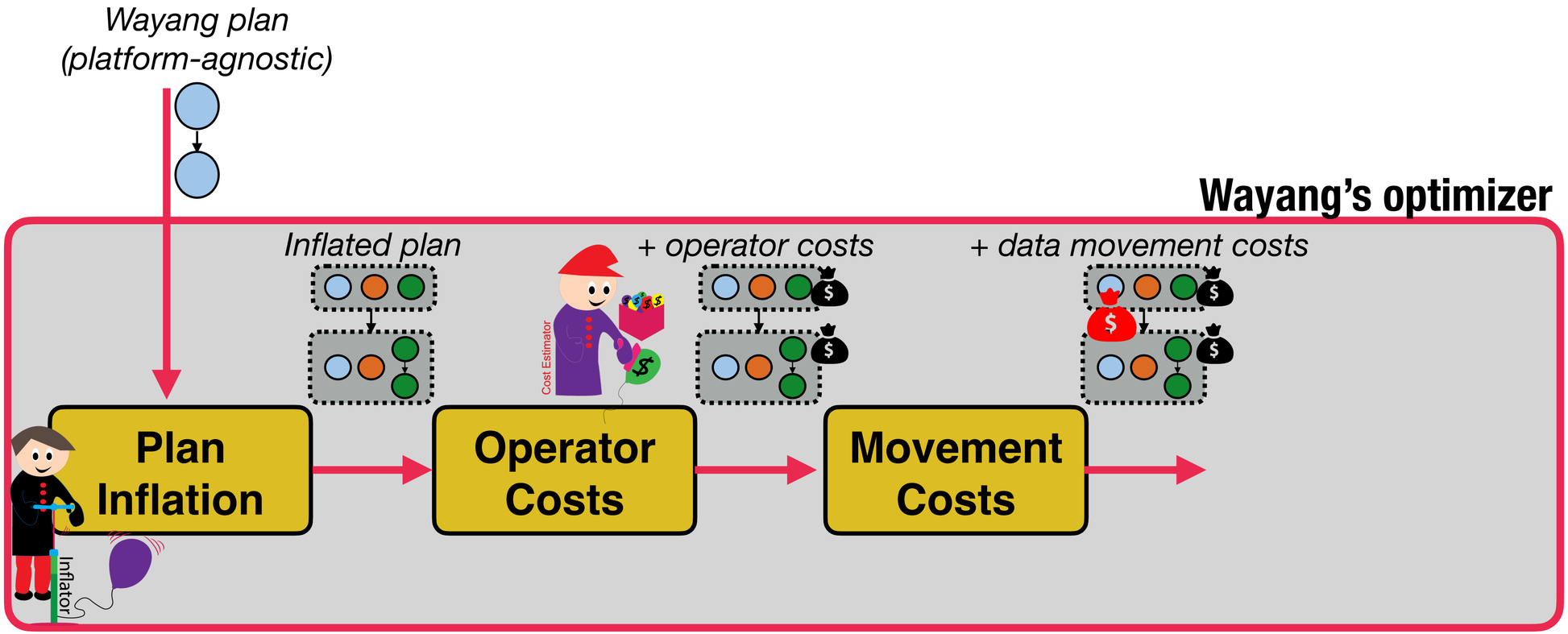


Plan enumeration

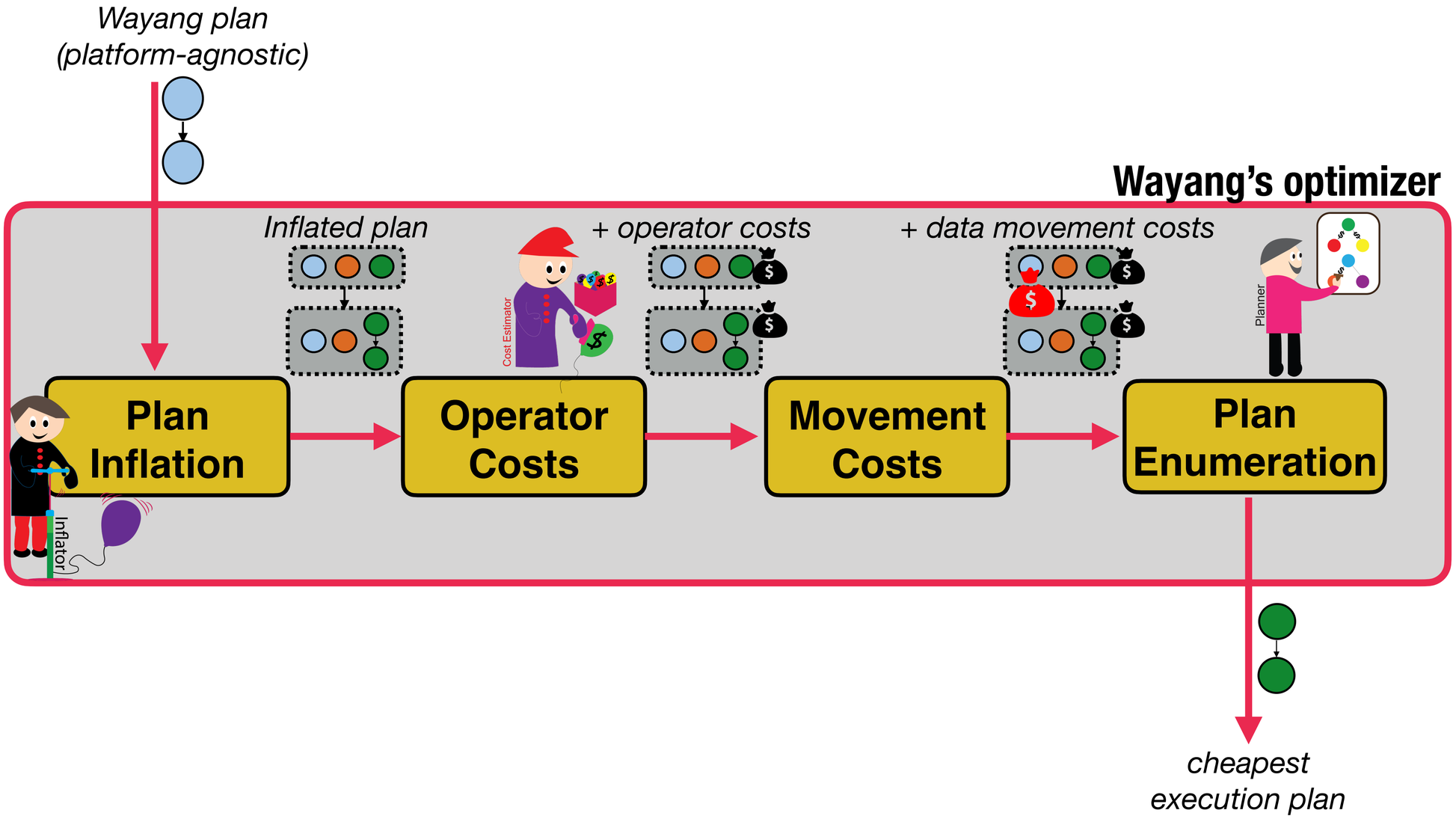


Lossless pruning

Cost-based QO in Apache Wayang



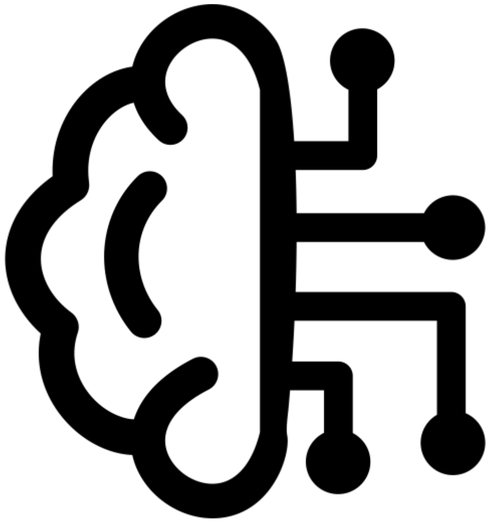
Cost-based QO in Apache Wayang



Unified Data Analytics Optimization

Cost-based

Rule-based



Learning-based

Unified Data Analytics Optimization

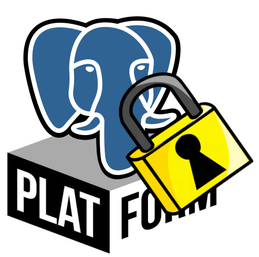
Cost-based

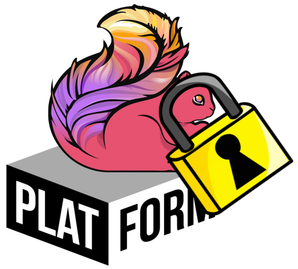
Rule-based

Learning-based

Problems with cost-based optimizer

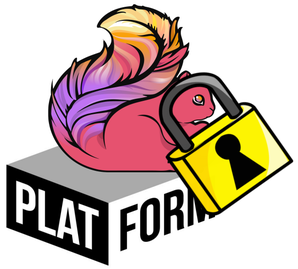
Problems with cost-based optimizer

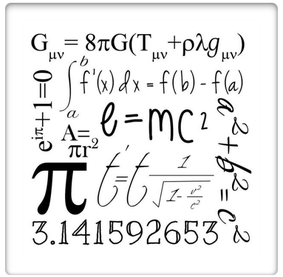




Low control

Problems with cost-based optimizer

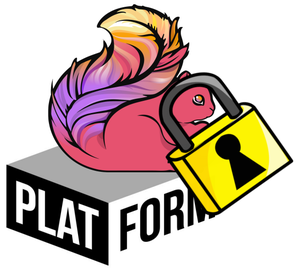


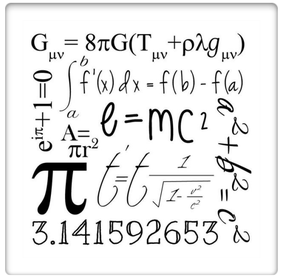


Low control

Tedious definition of cost formulas

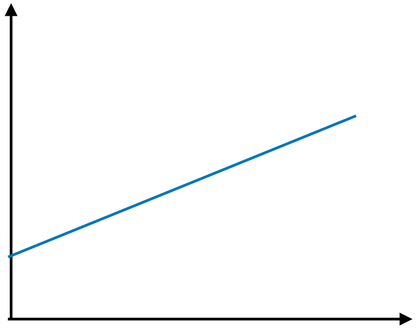
Problems with cost-based optimizer





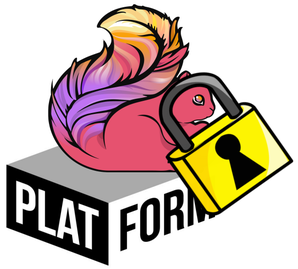
Low control

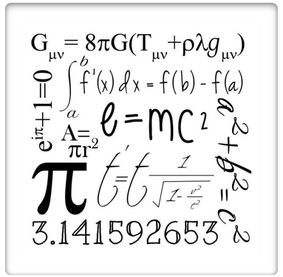
Tedious definition of cost formulas



Assume linear functions

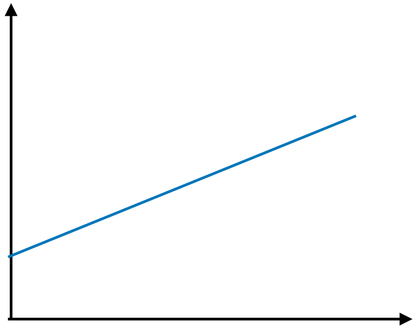
Problems with cost-based optimizer

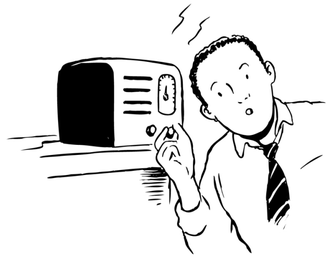




Low control

Tedious definition of cost formulas





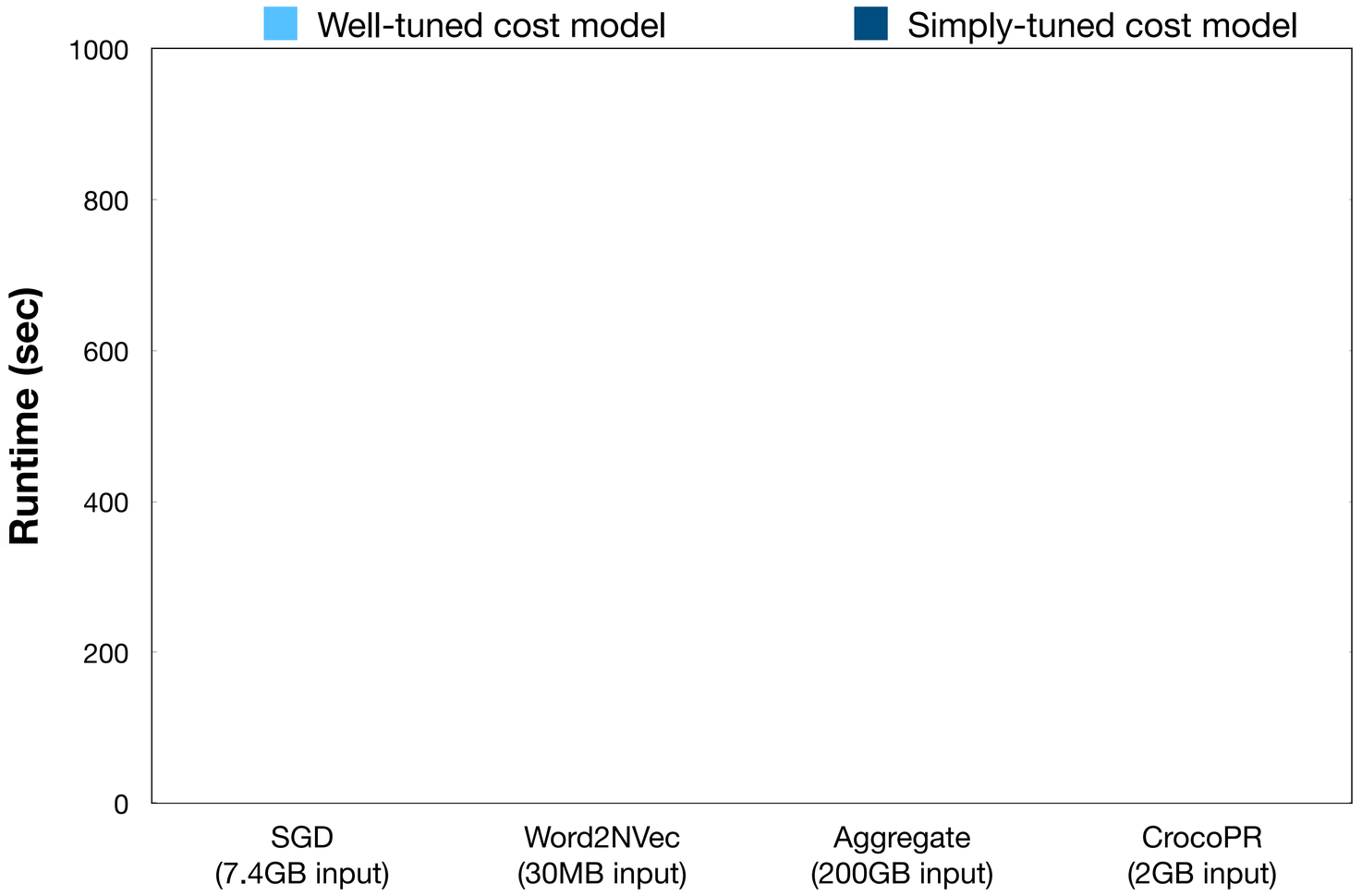
$$Cost_{sparkjoin} = a_1 x_{cpu} + a_2 x_{memory} + a_n x_{network} + \ldots$$

Assume linear functions

Hard to fine-tune

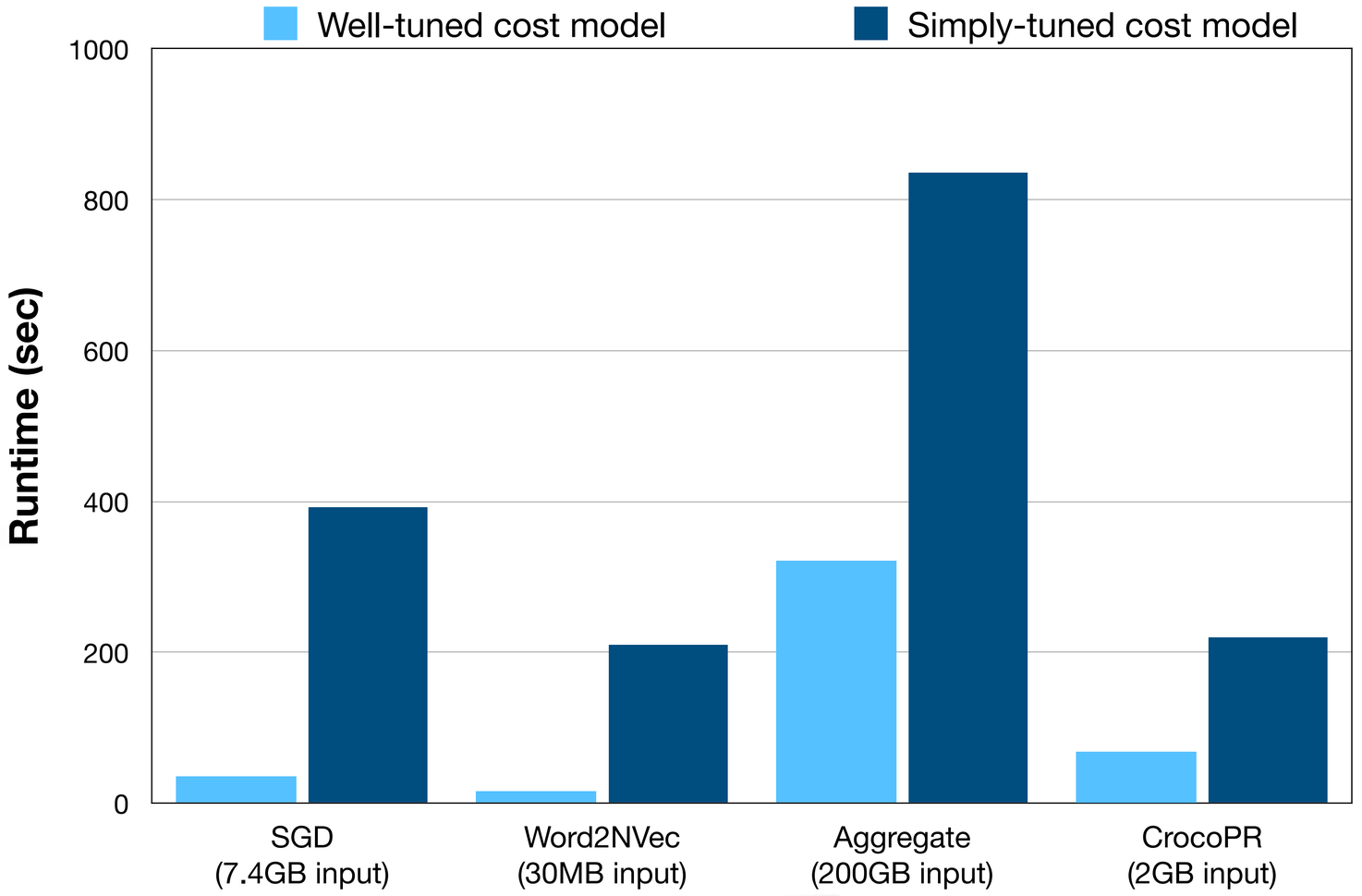
Effect of Cost Model Tuning

Using real cardinalities



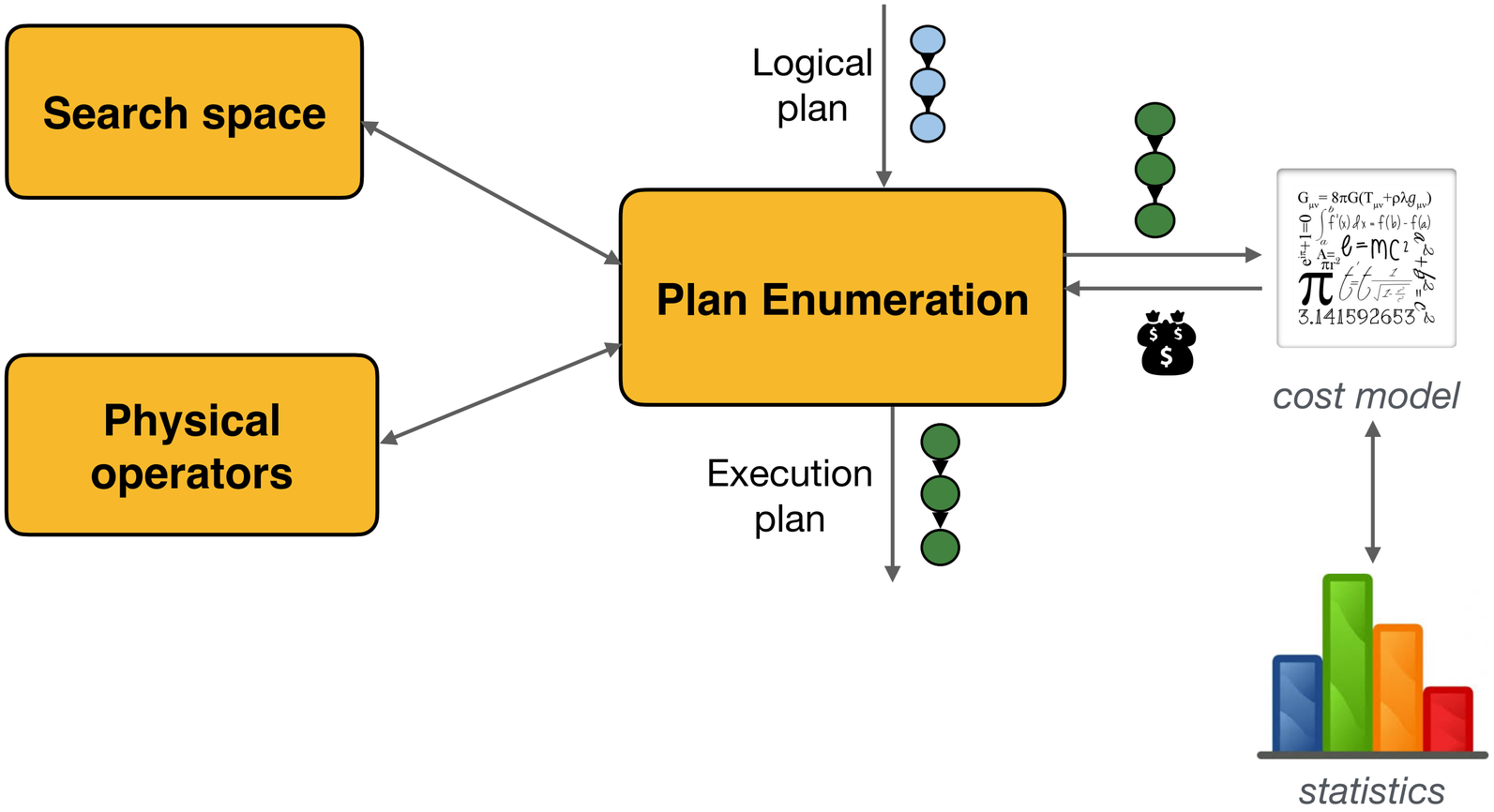
Effect of Cost Model Tuning

Using real cardinalities

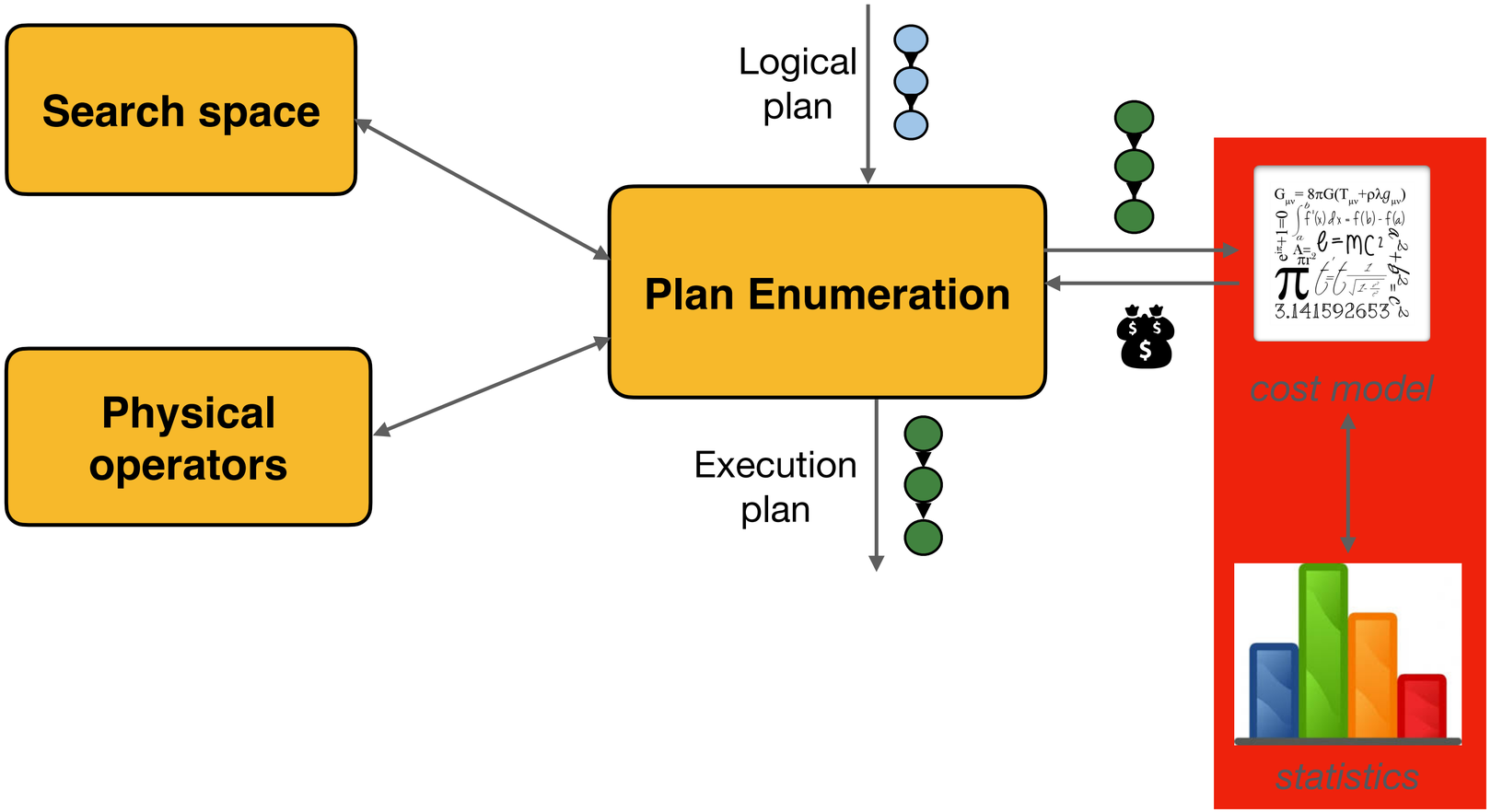


More than an order of magnitude better performance

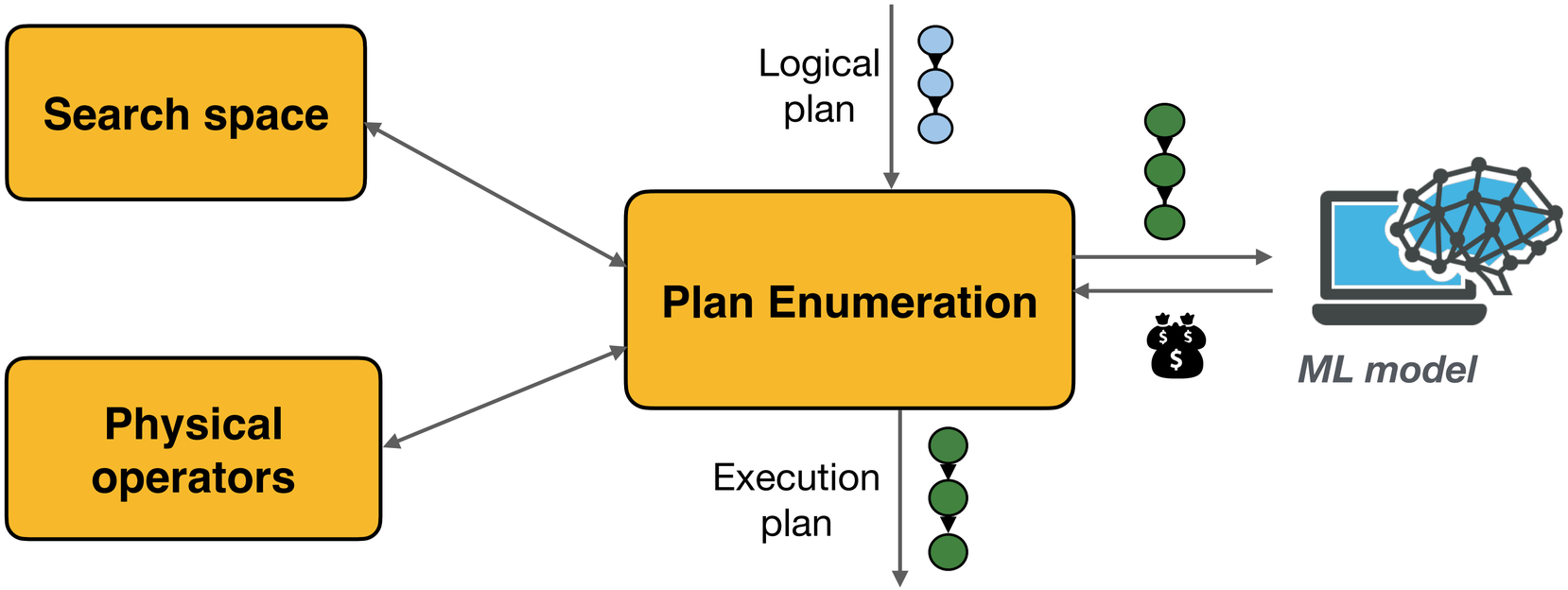
From Cost Model to ML Model



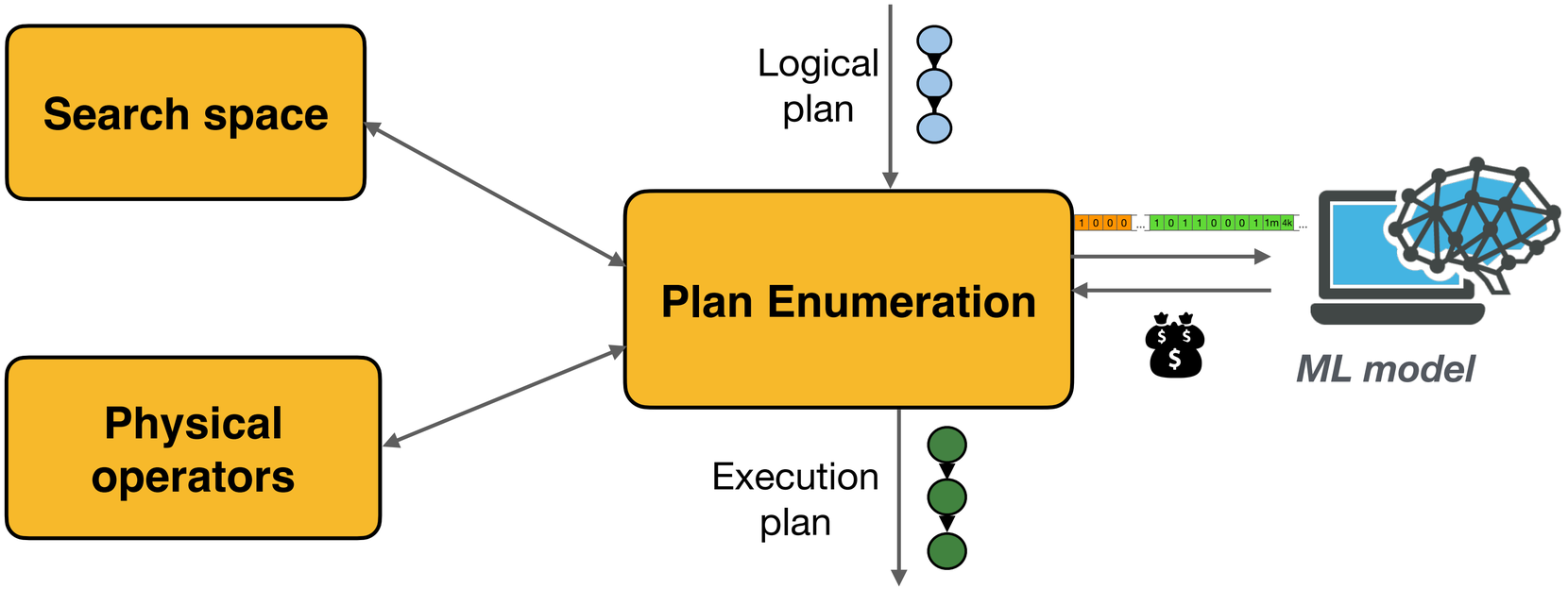
From Cost Model to ML Model



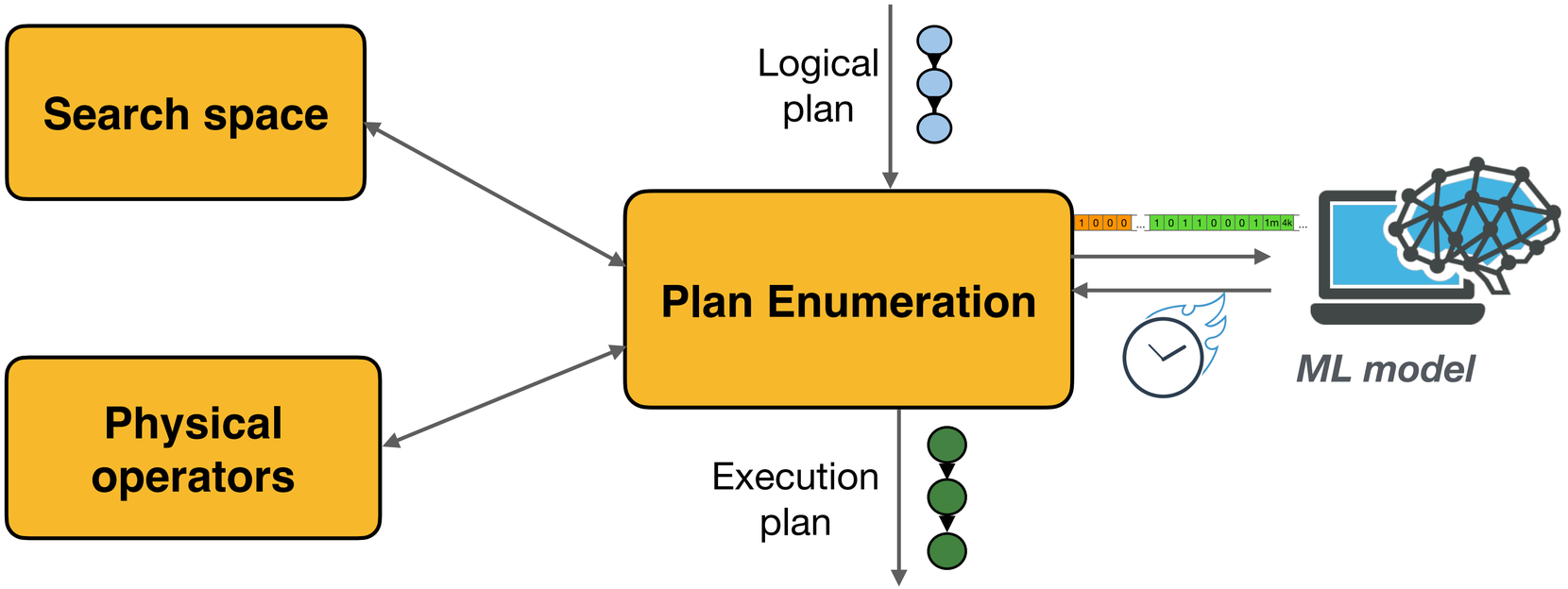
From Cost Model to ML Model



From Cost Model to ML Model



From Cost Model to ML Model



Challenges

1 Decoupling Applications

4 Extensibility

3

Automatic Cross-platform Data Analytics

2 Data Movement

Extensibility in Apache Wayang

Execution operators describing a platform

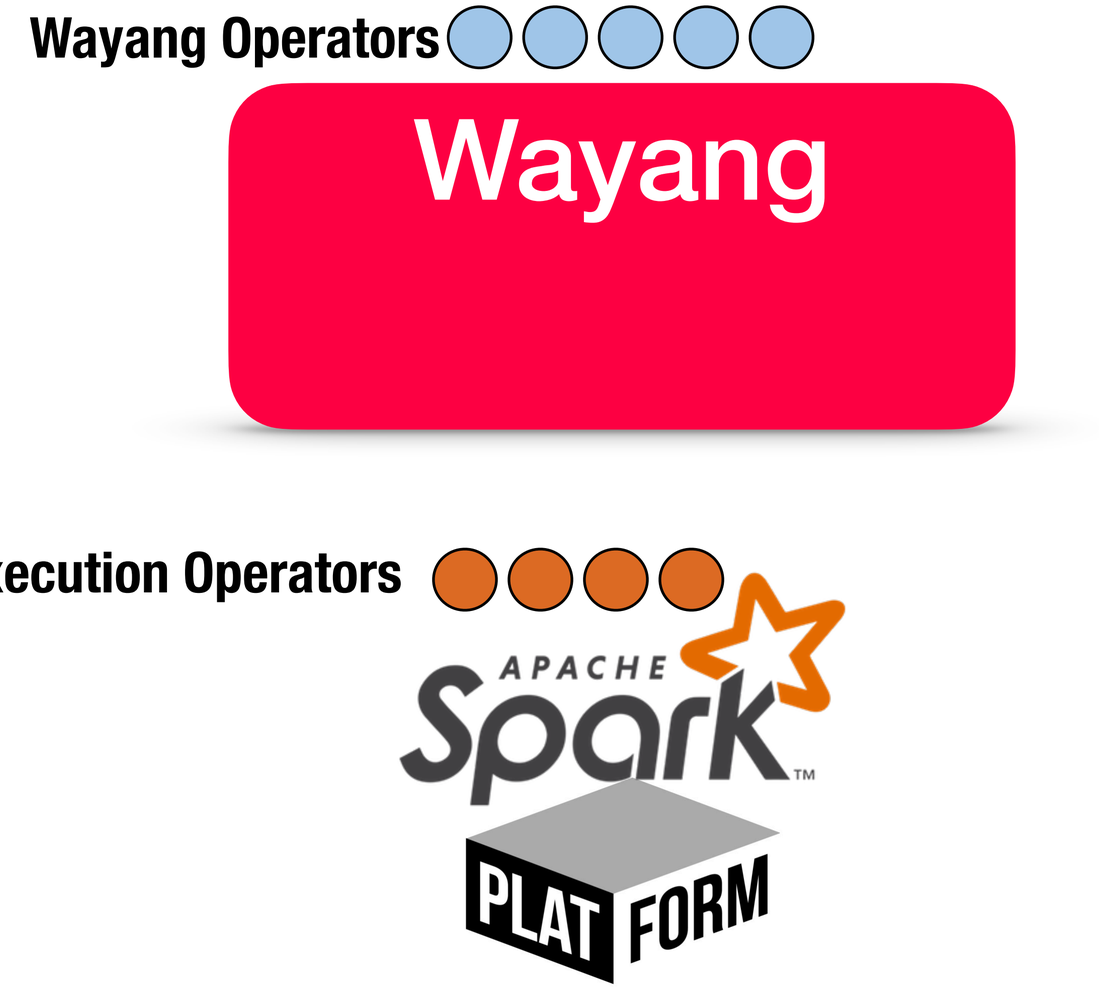
Wayang Operators

Wayang

RHEEMix in the data jungle: a cost-based optimizer for cross-platform systems. VLDB J. 29(6): 1287-1310 (2020)

Extensibility in Apache Wayang

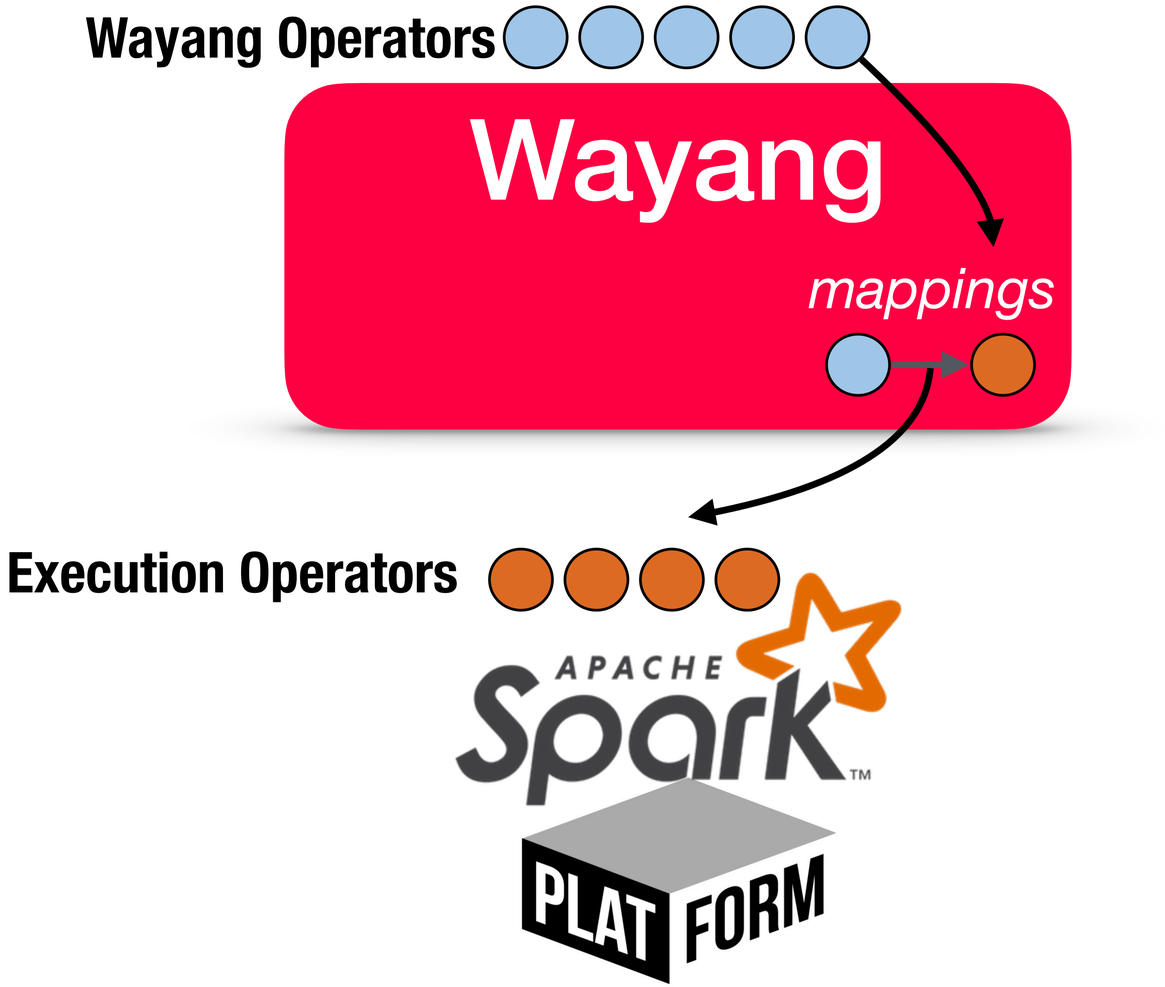
Execution operators describing a platform



RHEEMix in the data jungle: a cost-based optimizer for cross-platform systems. VLDB J. 29(6): 1287-1310 (2020)

Extensibility in Apache Wayang

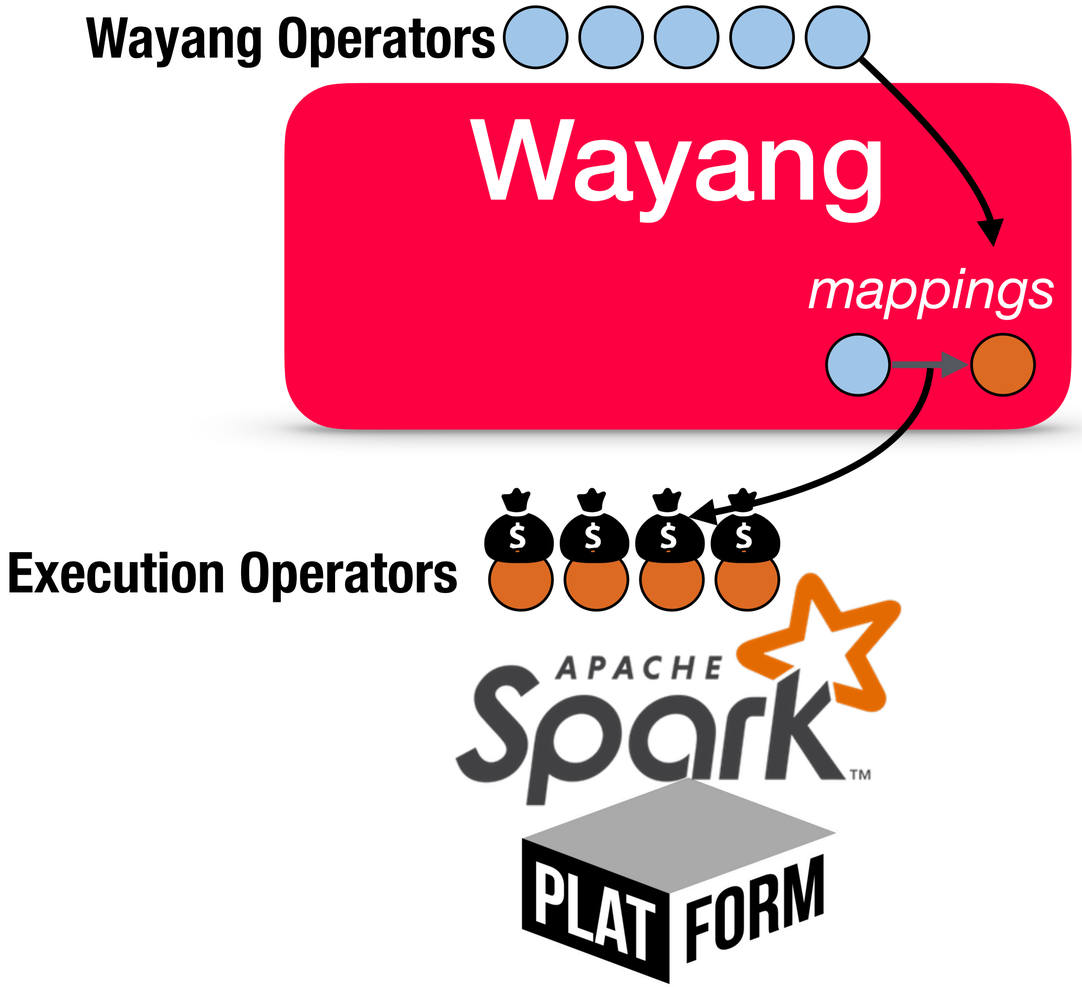
Execution operators describing a platform



 RHEEMix in the data jungle: a cost-based optimizer for cross-platform systems. VLDB J. 29(6): 1287-1310 (2020)

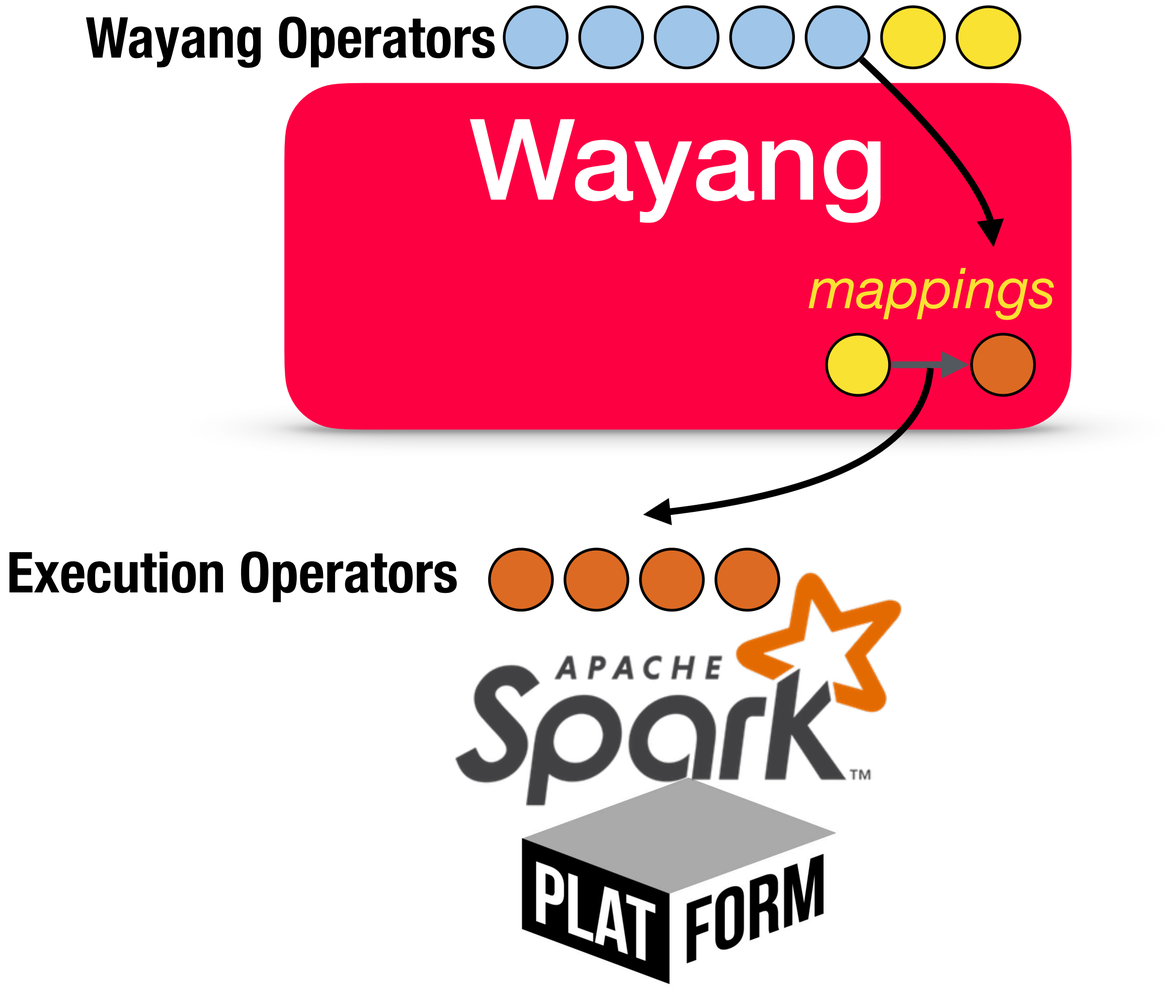
Extensibility in Apache Wayang

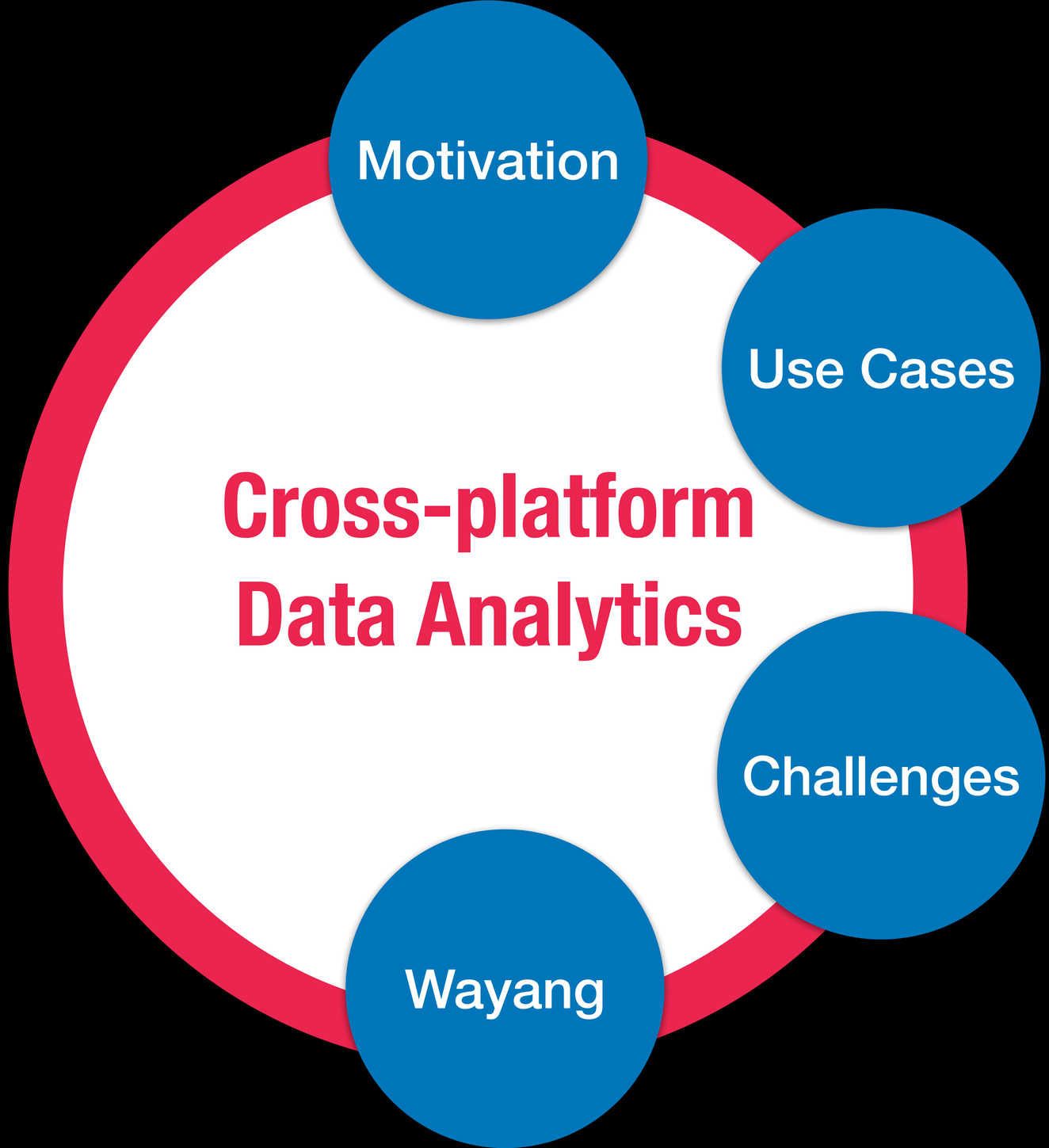
Execution operators describing a platform



 RHEEMix in the data jungle: a cost-based optimizer for cross-platform systems. VLDB J. 29(6): 1287-1310 (2020)

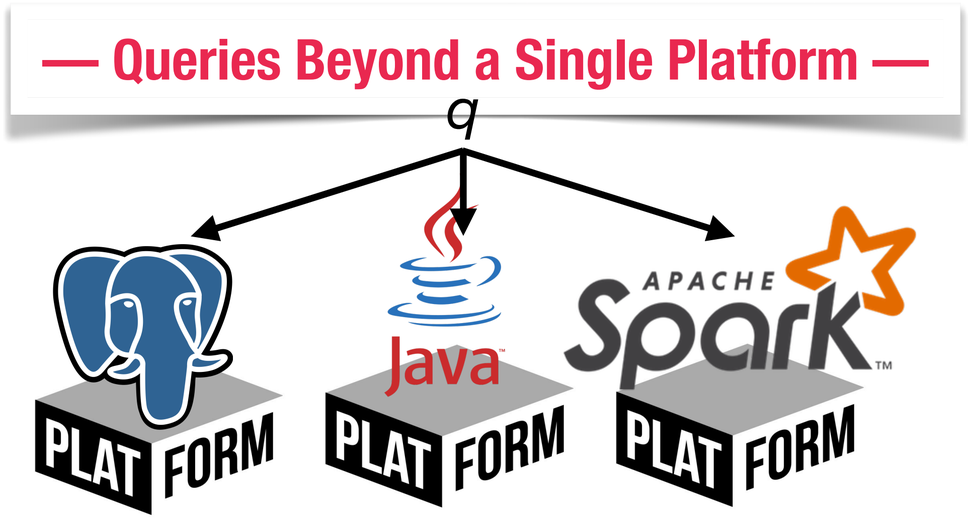
Extensibility in Apache Wayang

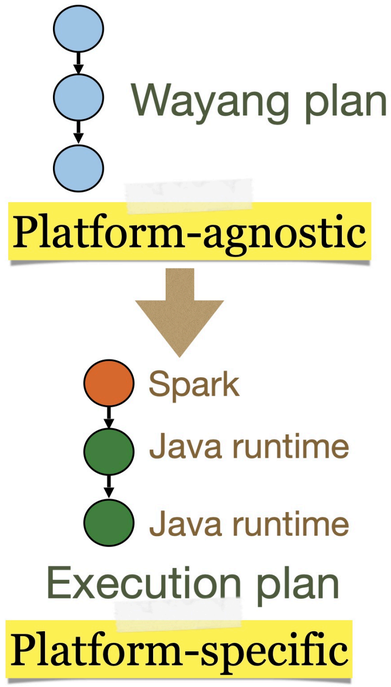


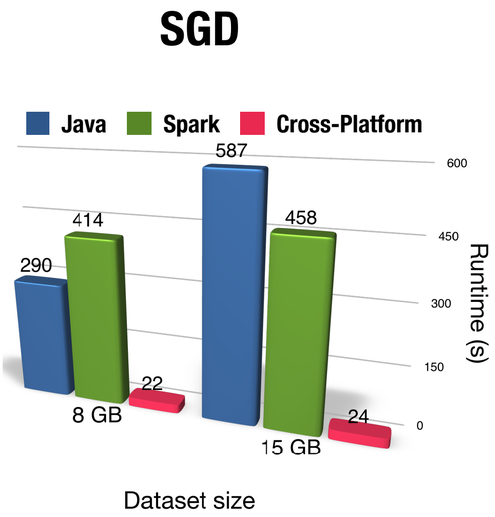

Motivation
Use Cases
Cross-platform
Data Analytics
Challenges
Wayang

Summary

real need for ***data processing independence***







https://wayang.apache.org/

https://github.com/apache/incubator-wayang

In the next lecture …

- After the break
- ML lifecycle with Maria

Readings

- K. Beedkar et al.: Apache Wayang: A Unified Data Analytics Framework, Sigmod Record (2023)
- D. Agrawal et al.: RHEEM: Enabling Cross-Platform Data Processing - May The Big Data Be With You! -. Proc. VLDB Endow. 11(11): 1414-1427 (2018)