SPARK+AI
SUMMIT EUROPE

FP&A with Spreadsheets and Spark

Oscar Castañeda-Villagrán
Universidad del Valle de Guatemala

#SAISEco2

About

- Researcher at Universidad del Valle de Guatemala.

- Research Interests:
  - Program Transformation,
  - Model-driven Data Product Design & Development

About

- Researcher at Universidad del Valle de Guatemala.

- Research Interests:
  - Program Transformation,
  - ~~Model driven Data Product Design & Development~~

About

- Researcher at Universidad del Valle de Guatemala.

- Research Interests:
  - Program Transformation,
  - ~~Model driven Data Product Design & Development~~
    - Design and Development of

About

- Researcher at Universidad del Valle de Guatemala.

- Research Interests:
  - Program Transformation,
  - ~~Model driven Data Product Design & Development~~
    - Design and Development of
    - Large-scale Data Products

About

- Researcher at Universidad del Valle de Guatemala.

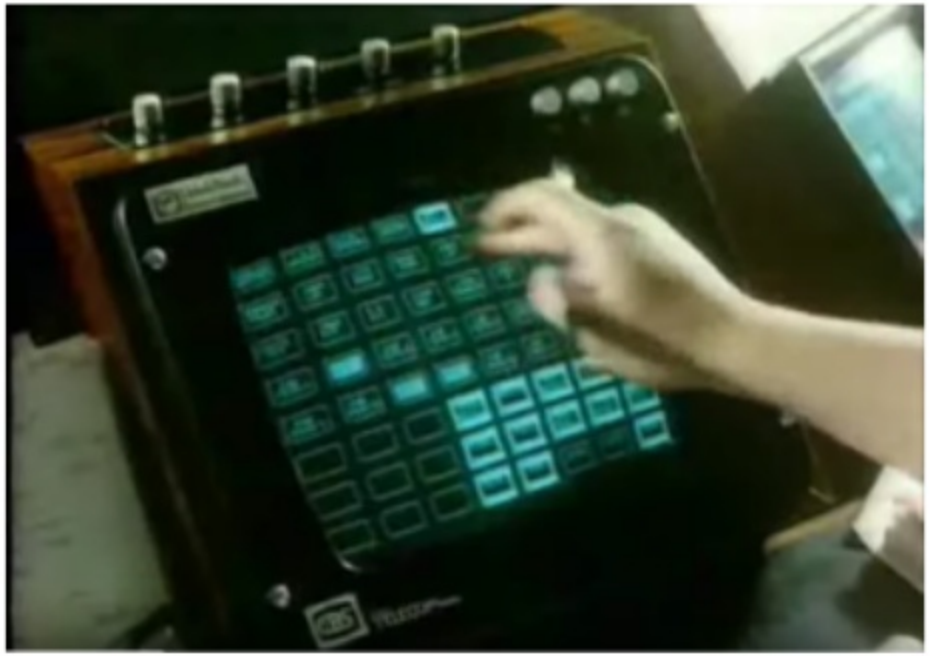
- Research Interests:
  - Program Transformation,
  - ~~Model driven Data Product Design & Development~~
    - Design and Development of
    - Large-scale Data Products
    - with Spreadsheets as Models

Prototyping …

http://bit.ly/2o5GmyY

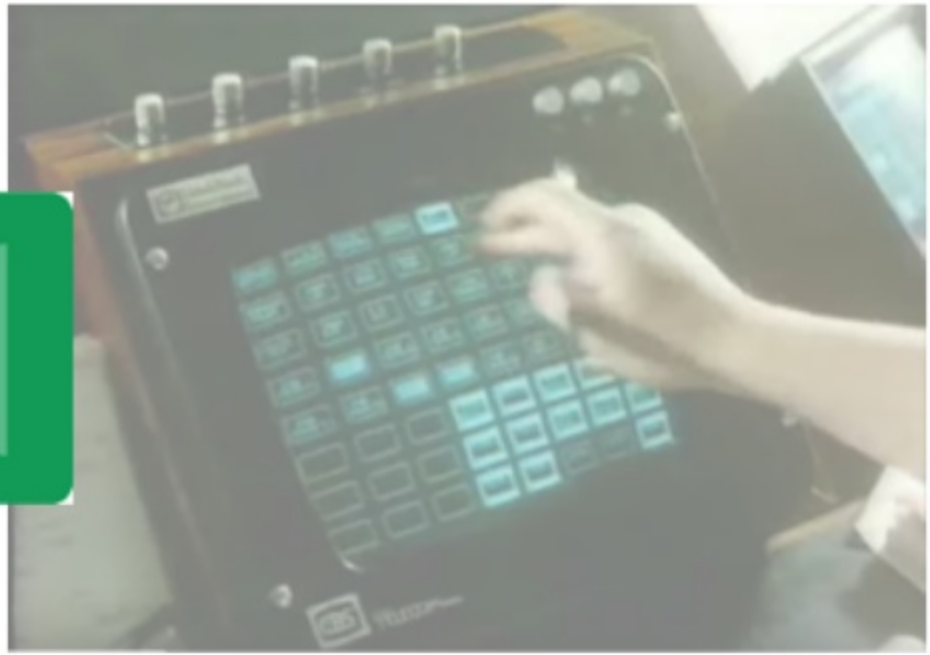
Prototyping Spark programs with …

http://bit.ly/2e5GmyY



http://bit.ly/2edYfMs

Spreadsheets!



http://bit.ly/2e5GmyY

http://bit.ly/2e0TZA8

http://bit.ly/2edYffMs


EXCEL

Agenda

- Problem Statement and Motivation
    - Architecture
- Program Transformation
    - Pipeline
    - Code-to-Code Transformation
- Code Generation
    - Abstract Tree
    - Parse Tree
- Spreadsheets as a DSL
    - Generating Code
- Demo
- Q&A

Disclaimer(s)

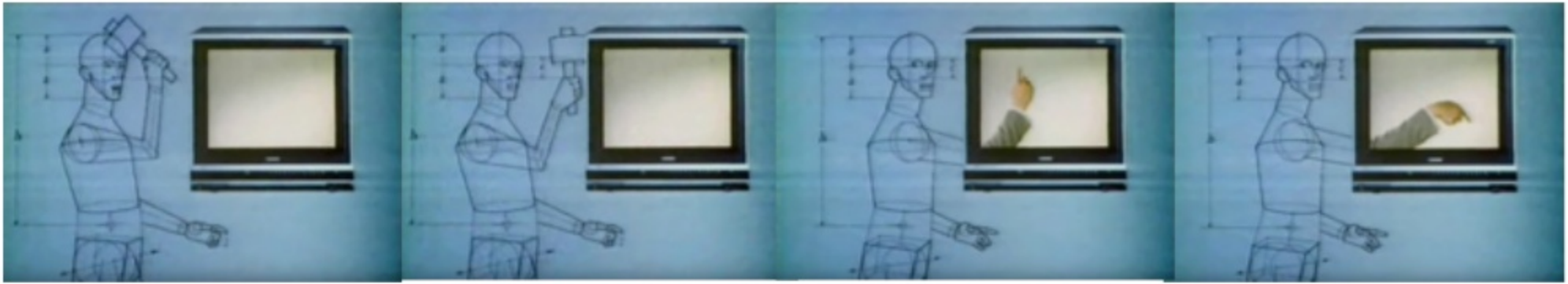
- Ongoing research …
- FP&A is one use case, but Spreadsheets are much broader!

Disclaimer(s)

- Ongoing research …
- FP&A is one use case, but Spreadsheets are much broader!
    - E.g. People have even modeled Turing machines with Spreadsheets! [1]

Problem Statement

Prototype FP&A programs using Spreadsheet formulas and automatically translate to Scala / Spark.



Problem Statement

Prototype ***Any*** program using Spreadsheet formulas and automatically translate to Scala / Spark.

Motivation

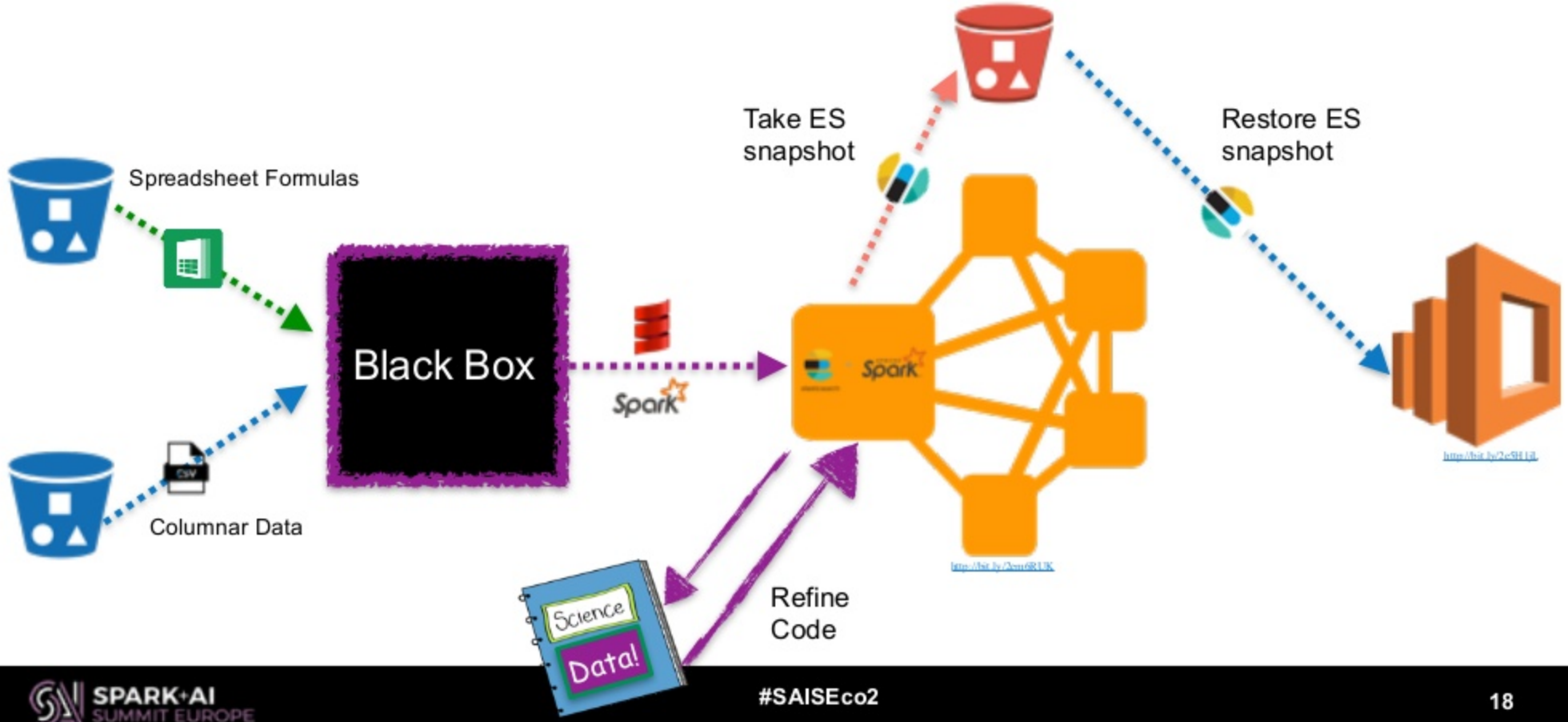
- At Spark Summit Europe 2016 I presented the **Sparksheet** code generator for Spreadsheet formulas.
- Initially **Sparksheet** supported only 5 Spreadsheet formulas, now it supports 150+ Spreadsheet formulas!
- Motivation is finding use cases.

Motivation

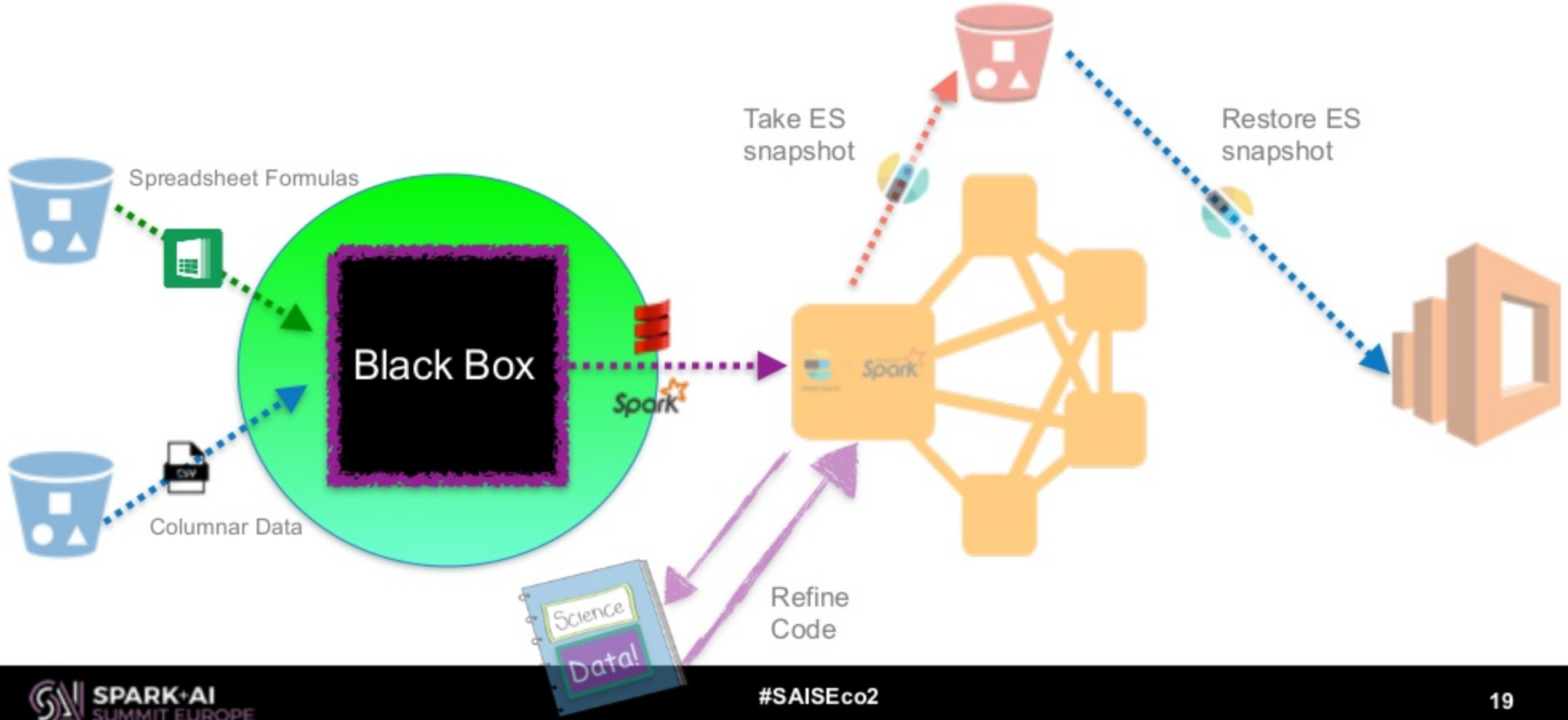
Automatically translate
Spreadsheet datasets* to
Spark data pipelines on
Scala/Spark

*Spreadsheet dataset = Structured data + Spreadsheet formulas

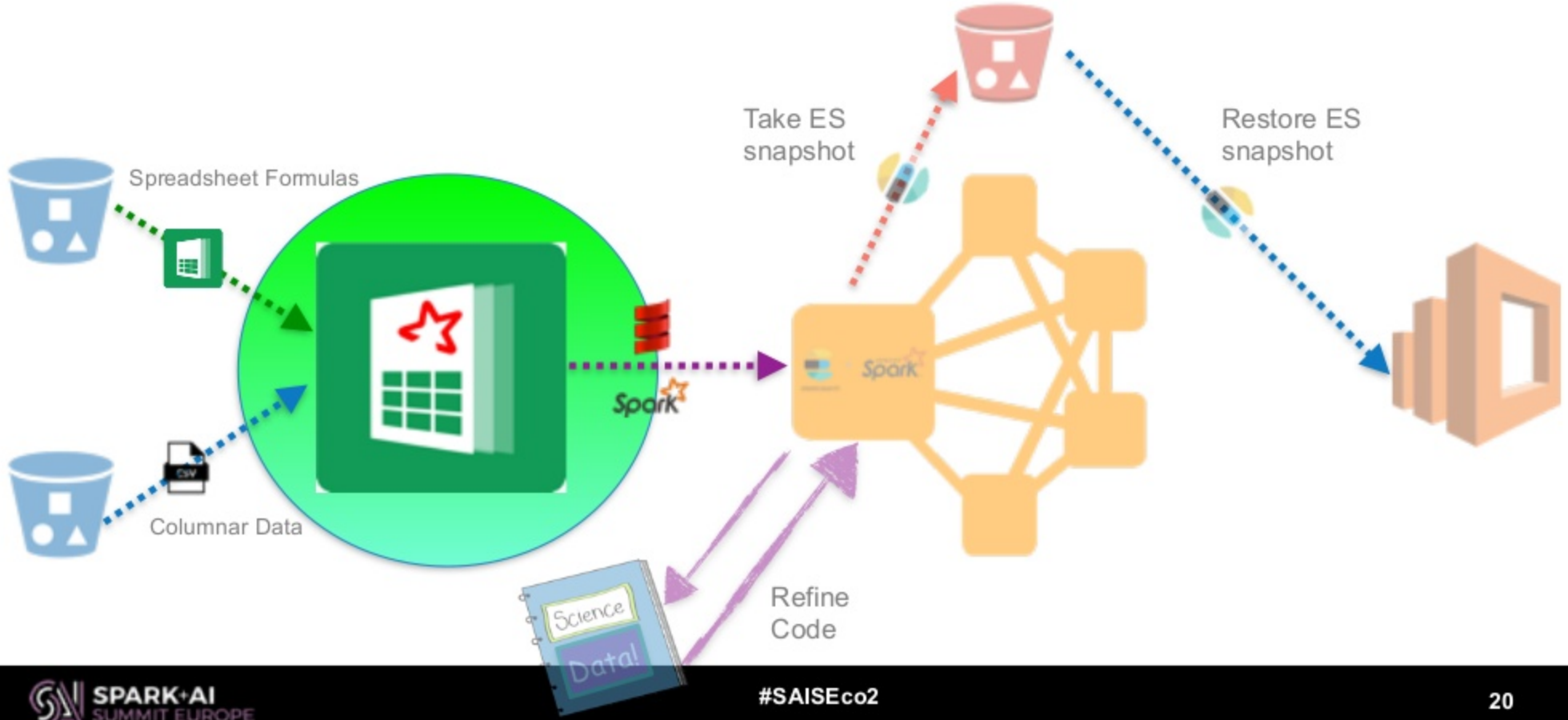
Architecture



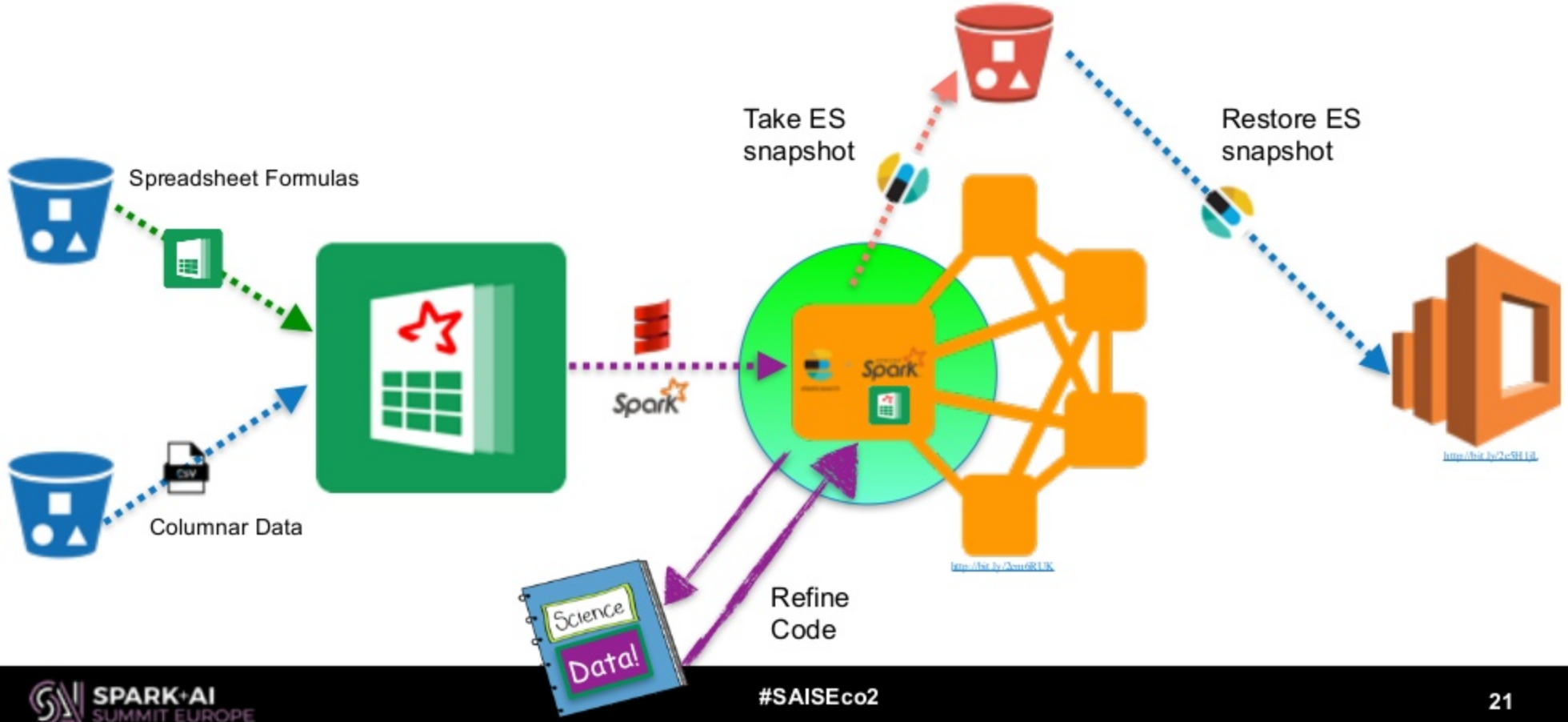
Architecture



Architecture



Architecture



Program Transformation

"A **program transformation** is any operation that takes a computer program and generates another program."

https://en.wikipedia.org/wiki/Program_transformation

Program Transformation Pipeline

http://bit.ly/2e0TZA8

http://bit.ly/2cfajbl

http://bit.ly/2dl0cFg

Demo #1

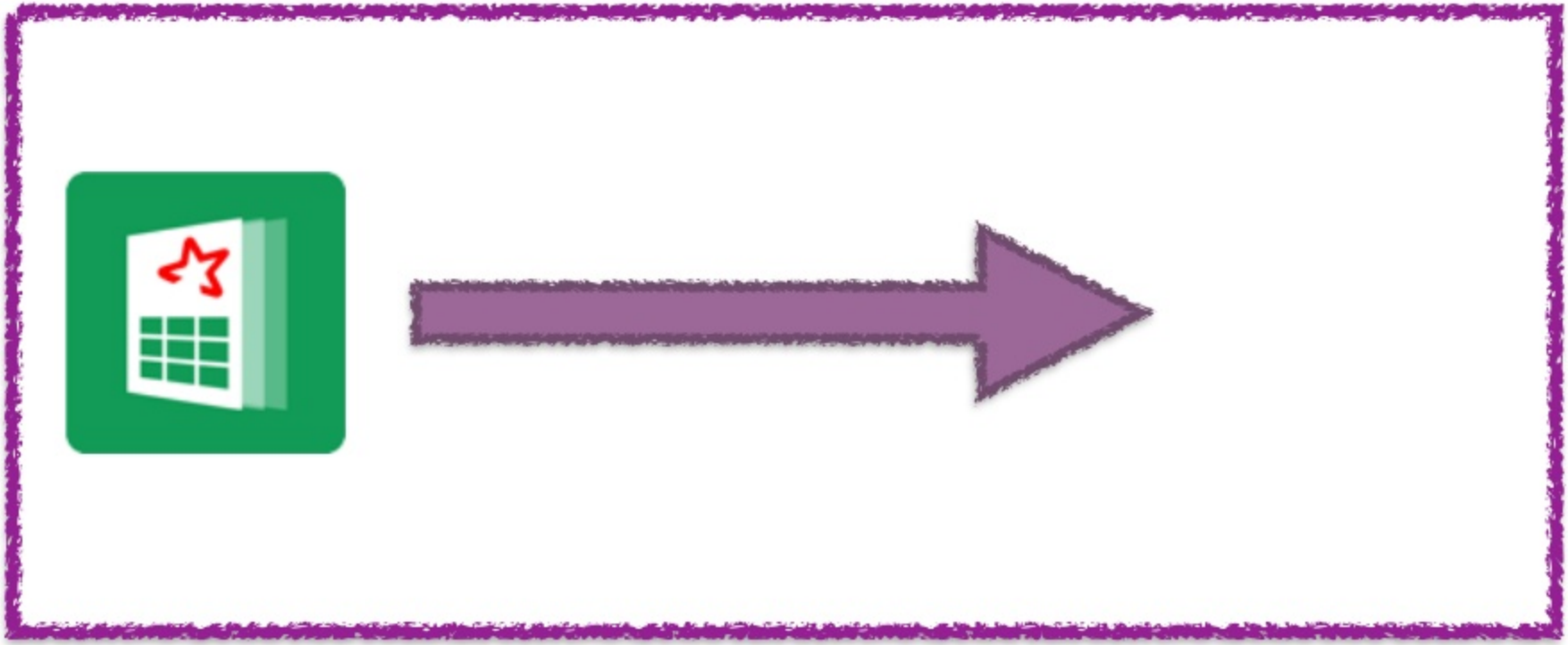
http://bit.ly/2e0TZA8

1. Show Spreadsheet model
2. Show Complex Spreadsheet Formula

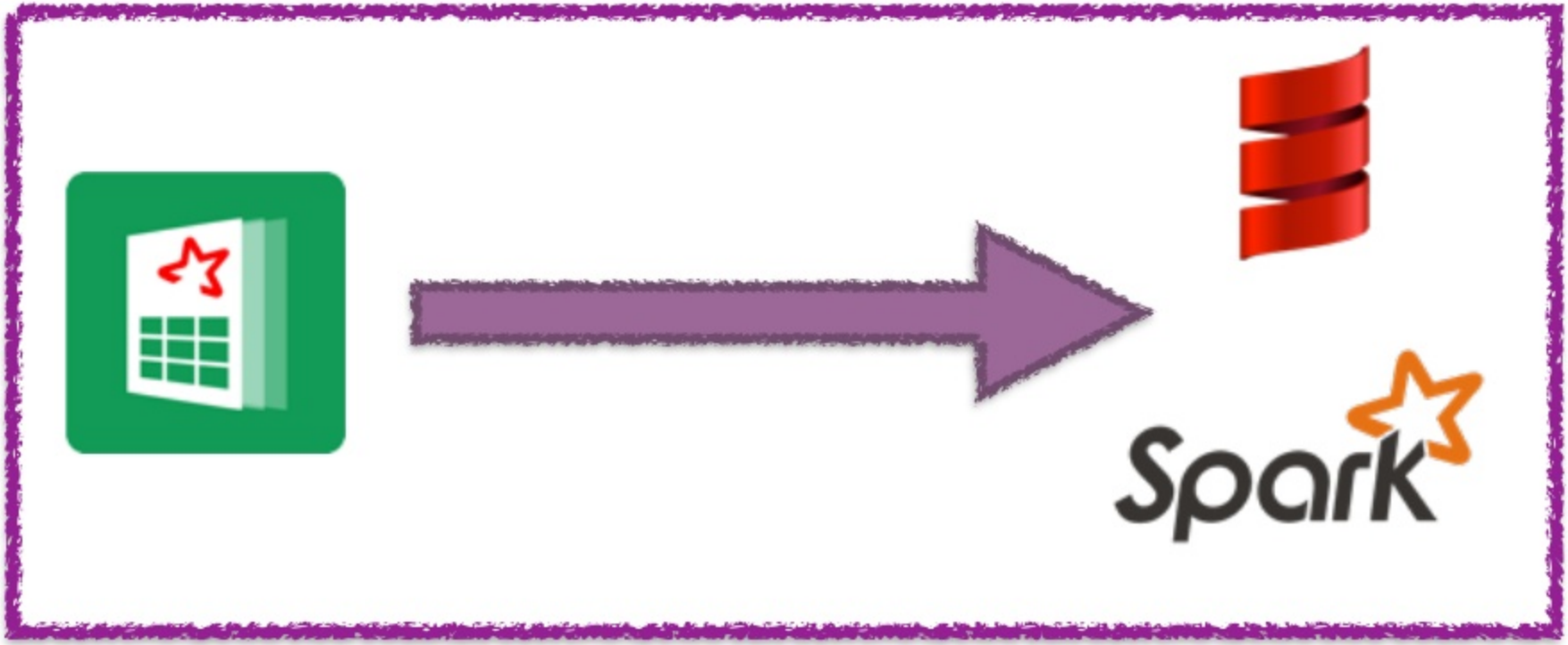
Program Transformation

Source program in Excel formula language.

Demo #2



Demo #2



Demo #2

Sparksheet

Complex

Spreadsheet Formula

Spark

SparkSheet Demo | SparkSheet - Databricks

localhost:39901

SparkSheet

Input Excel Formula

```
IF(A23<=16,0,COUNT(IF(COUNTA(OFFSET
($A3,0,
(MATCH(A03,$B$2:$AC$2)),1,2))>1,1,0
),IF(COUNTA(OFFSET($A3,0,
(MATCH(A03,$B$2:$AC$2))+1,1,2))>1,1
```

Convert to Scala | 1.2.0

Demo

Output Spark App in Scala

```
import org.apache.spark.sql._
import org.apache.spark.sql.functions._
import scala.util.Try

//******************
//The following code was generated for this spreadsheet formula:
//******************



val ifColumn22DF = ifColumn1(DFname.coalesce(1), "1", "lit(A23<=16)", "lit(0)",
"if_22", "COLUMN", "input_ifColumn22", "output_ifColumn22")
val if_22 = countColumn4(DFname.coalesce(1), "1",  "count_4", "count_5", "count_6",
"count_7", "count_8", "count_9",  "input_countColumn4", "output_countColumn4")
val count_4 = ifColumn1(DFname.coalesce(1), "1", "if_23>1", "lit(1)", "lit(0)",
"COLUMN", "input_ifColumn23", "output_ifColumn23")
val if_23 = countaColumn19(COUNTA.coalesce(1), "1",  "counta_19",
"input_countaColumn19", "output_countaColumn19")
val counta_19 = offsetColumn19(DFname.coalesce(1), "1",  A3, "lit(0)", "offset_19",
"lit(1)", "lit(2)",  "input_offsetColumn19", "output_offsetColumn19")
val offset_19 = matchColumn19(DFname.coalesce(1), "1", "A03", "B2", "AC2", "COLUMN",
"input_matchColumn19", "output_matchColumn19")

val count_4 = ifColumn1(DFname.coalesce(1), "1", "if_24>1", "lit(1)", "lit(0)",
"COLUMN", "input_ifColumn24", "output_ifColumn24")
val if_24 = countaColumn20(COUNTA.coalesce(1), "1",  "counta_20",
"input_countaColumn20", "output_countaColumn20")
val counta_20 = offsetColumn20(DFname.coalesce(1), "1",  A3, "lit(0)", "offset_20",
"lit(1)", "lit(2)",  "input_offsetColumn20", "output_offsetColumn20")
val offset_20 = matchColumn20(DFname.coalesce(1), "1", "A03", "B2", "AC2", "COLUMN",
"input_matchColumn20", "output_matchColumn20")

val count_4 = ifColumn1(DFname.coalesce(1), "1", "if_25>1", "lit(1)", "lit(0)",
"COLUMN", "input_ifColumn25", "output_ifColumn25")
val if_25 = countaColumn21(COUNTA.coalesce(1), "1",  "counta_21",
"input_countaColumn21", "output_countaColumn21")
```

Code-to-Code Transformation

Code-to-Code Transformation

"The input to the code generator typically consists of a **parse tree** or an **abstract syntax tree**."

http://bit.ly/2dH0ybF

https://en.wikipedia.org/wiki/Code_generation_(compiler)

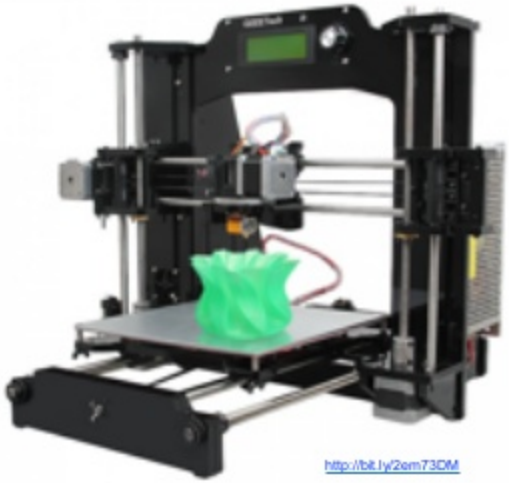
Generating Code

"An elegant way to generate code from an AST is to write a class for each non-terminal node in the tree, and then each node in the tree simply generates the piece of code that it is responsible for."

http://www.codeproject.com/Articles/26975/Writing-Your-First-Domain-Specific-Language-Part

Generating Code

A **practical** way to generate code is to take a Parse Tree and write a pretty printer for the target language.



http://bit.ly/2em73DM

Generating Code (Example)

SUM(A,C)

Generating Code (Example)

SUM(A,C)

```
import org.apache.sql._
import org.apache.spark.sql.functions._

def SUM(A:Int, C:Int): Int = {
  return A + C
}
val applySUM = udf(SUM _)
val sumDF = baseDF.withColumn("",
            applySUM(
              col("A"),
              col("C")
            ))
```

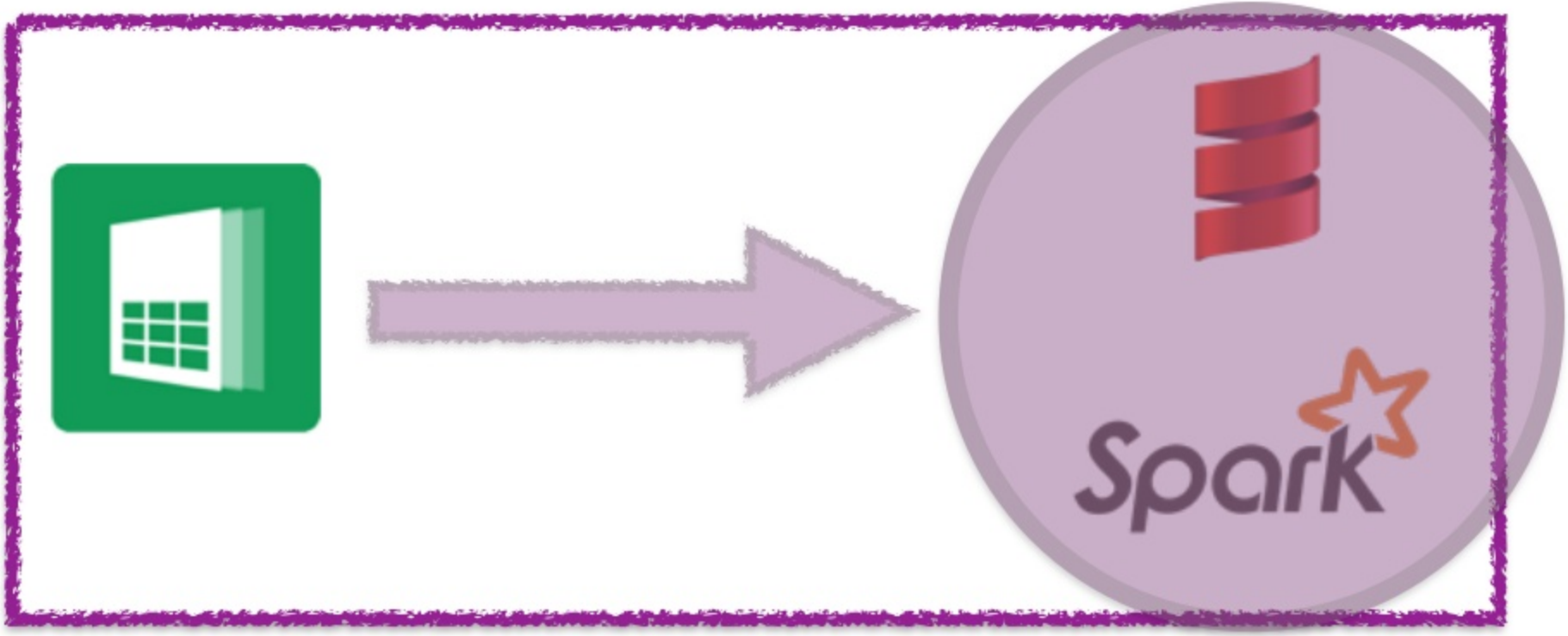
Generating Code (Example)

SUM(A,C)

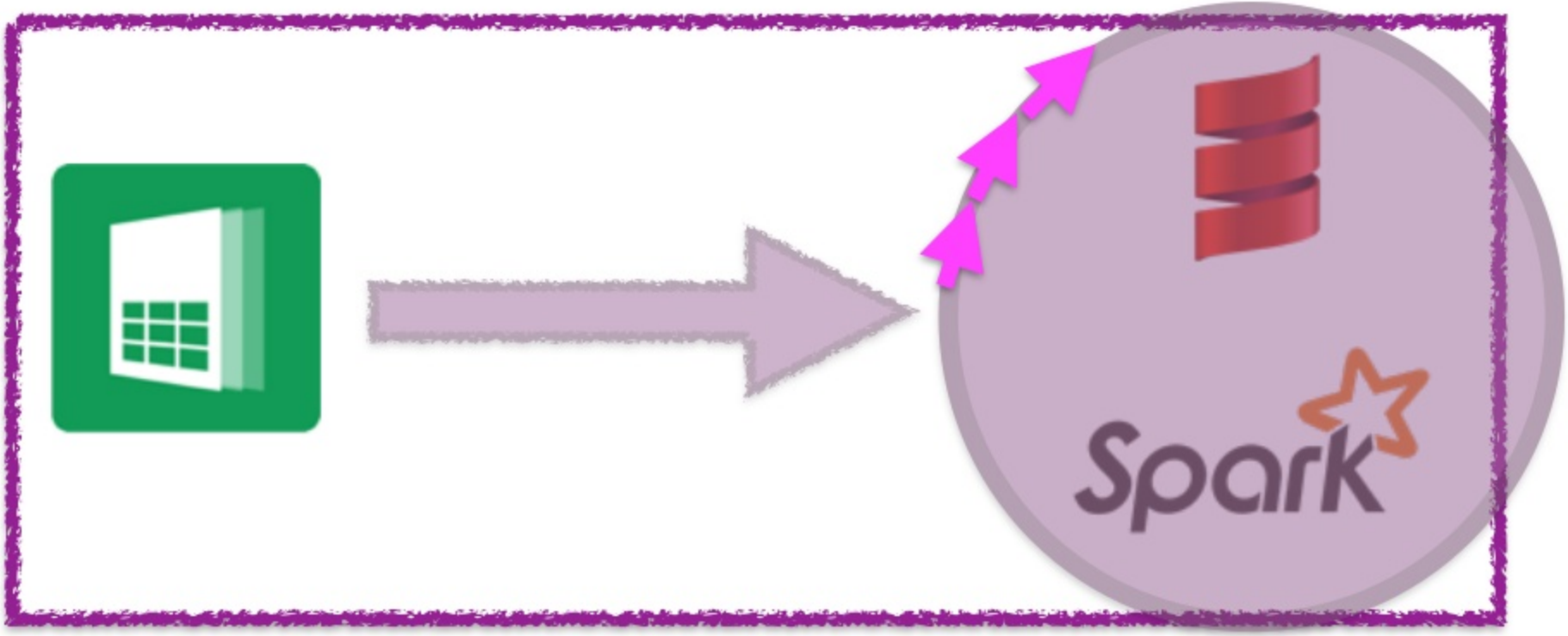
```
import org.apache.sql._
import org.apa rk.sql.functions._

def SUM A:In C:Int Int = {
  return A +
}
val applySU aseDF.w )
val sumDF lumn("",
        applySUM(
          col("A"),
          col("C")
        ))
```

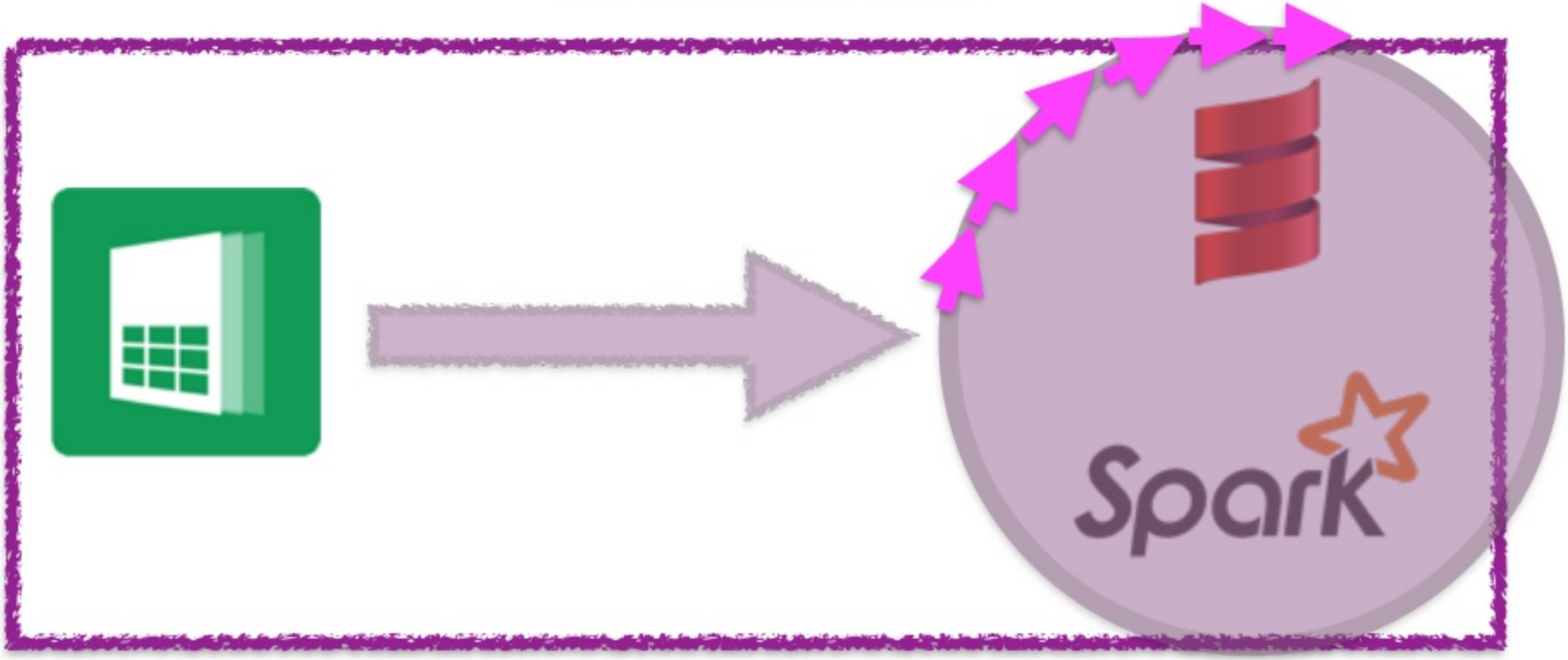
Demo #3



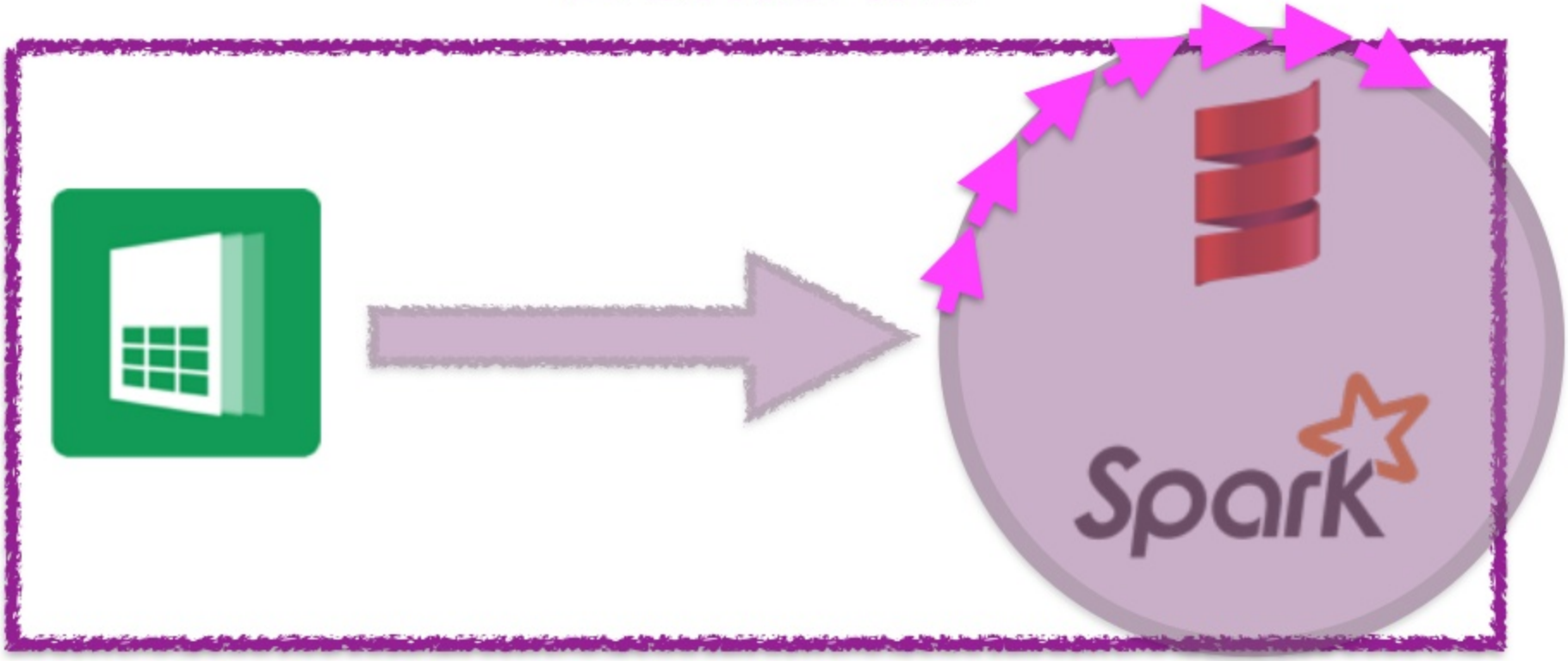
Demo #3



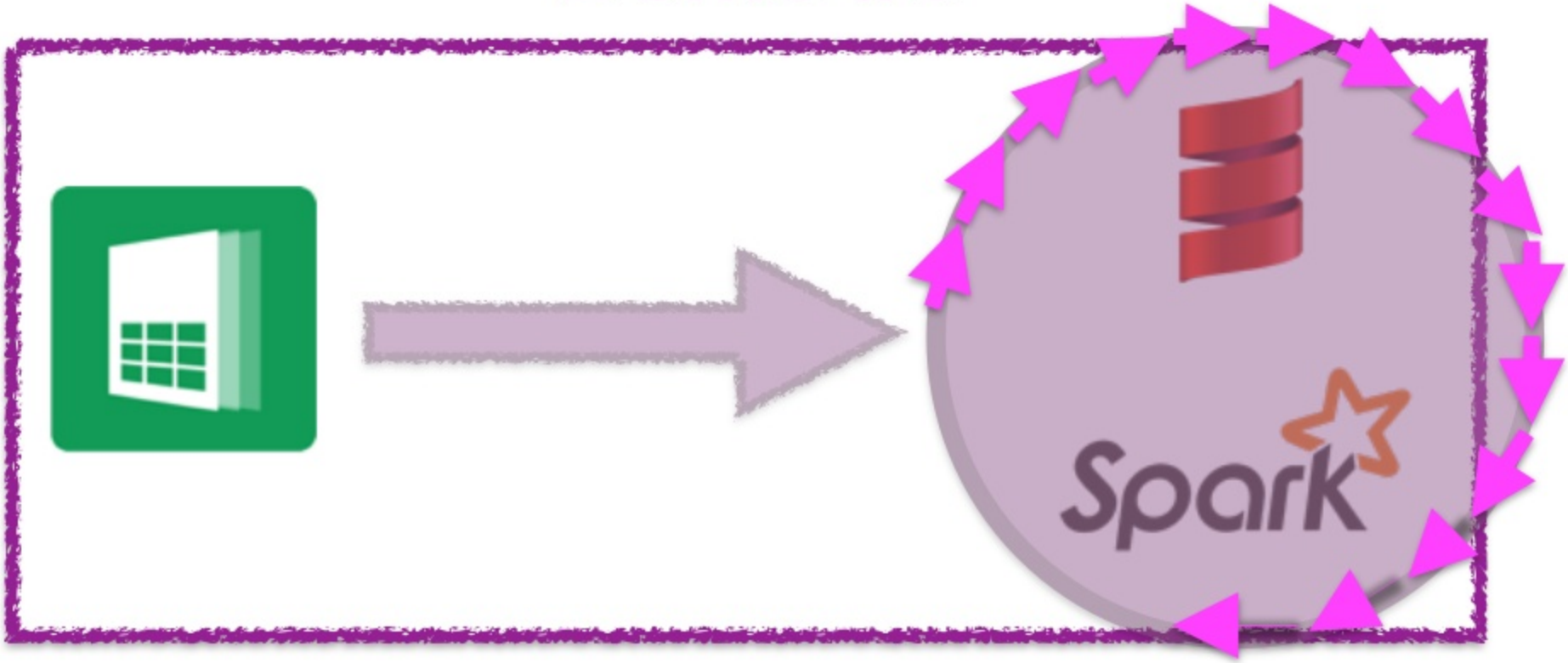
Demo #3



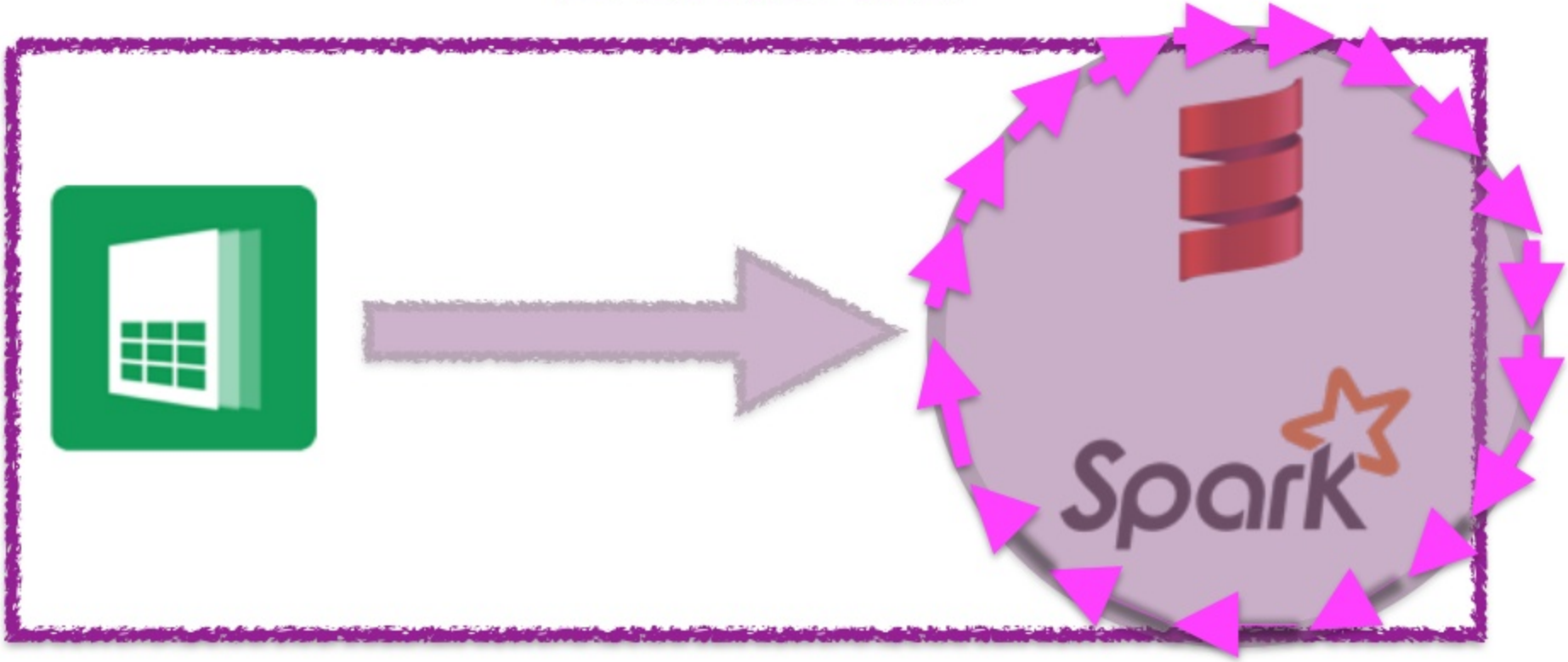
Demo #3



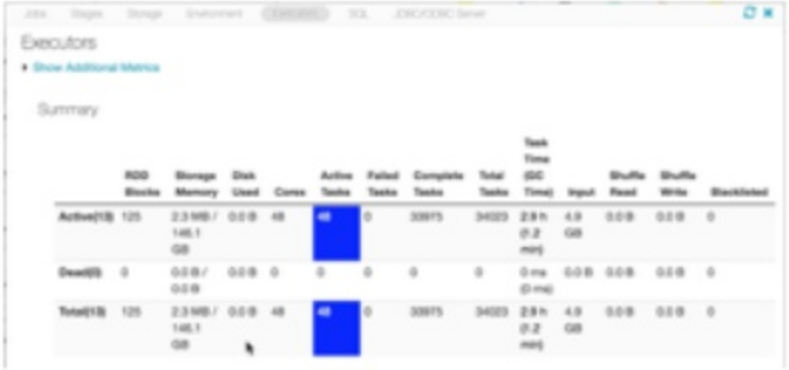
Demo #3

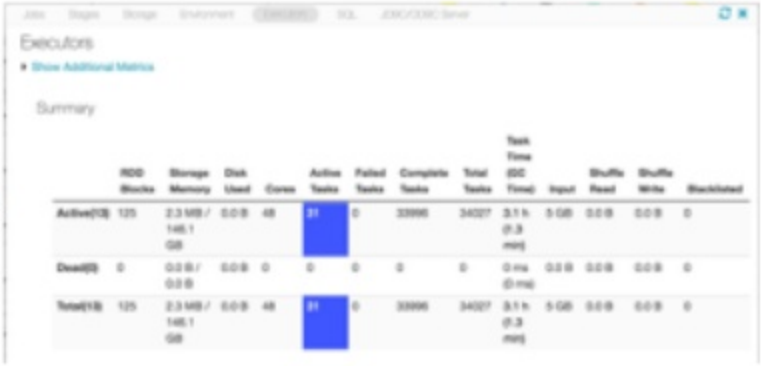


Demo #3

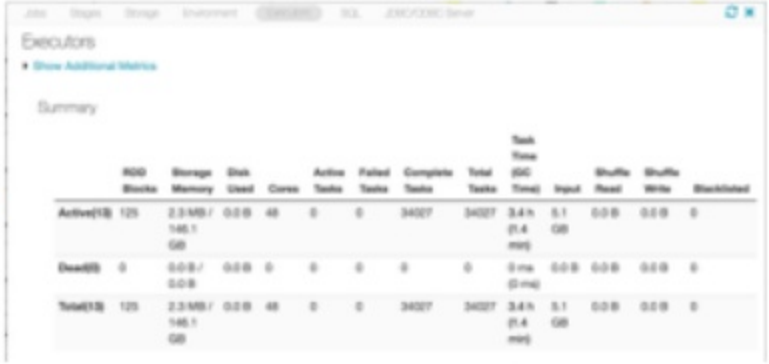


Demo #3


Executors
Show Additional Metrics
Summary


Executors
Show Additional Metrics
Summary


Executors
Show Additional Metrics
Summary


Executors
Show Additional Metrics
Summary

Spreadsheets as a DSL

- Spreadsheet is a powerful data modeling tool.

- Start simple and evolve into a complex ML pipeline.

- Spreadsheets are suitable to many domains (FP&A is one such domain).

What have we seen?

- Spreadsheet applications as Prototypes for Spark programs
- Program Transformation
  - How to model as Pipeline
  - Why considered Code-to-Code Transformation
- How to Generate Code
  - AST (elegant)
  - Parse Tree (practical)
- Spreadsheets as a DSL
  - Generating Code
- Next Steps

Next Steps

- Use cases!
- Modeling Machine Learning in a Spreadsheet
- Prototype D|'s and ML|'s in a Spreadsheet

References

- A Grammar for Spreadsheet Formulas Evaluated on Two Large Datasets – Efthimia Aivaloglou, David Hoepelman & Felienne Hermans, Proceedings of SCAM '15

- http://www.felienne.com/archives/2974

- Pictures in presentation from Boards of Canada video "roygbiv"

  https://youtu.be/yT0gRc2c2wQ

Q&A

SPARK+AI
SUMMIT EUROPE

THANK YOU.

Email: ofcastaneda@uvg.edu.gt
Twitter: @oscar_castaneda