Bulletproof Jobs @ Scale
Patterns for Spark Magic

Simeon Simeonov
Founder & CTO, Swoop, @simeons

Bulletproof jobs

Reliable operations
over long periods of time
and fast recovery from failures

petabyte scale data engineering & ML
run our business

Spark vs. magic

great magic comes
from the consistent application
of smart conventions

(often quite easy to add on top of Spark)

Problem: job failure response

Your job failed.

What should you do?

90+% of job failures are transient

Does that help?

If I had a magic wand…

I'd like my system to auto-retry the job.

Would that be OK?

Automatic retry works only for idempotent jobs

(if multiple executions don't change state)

Idempotence in action

```
DROP TABLE IF EXISTS my_table
```

Idempotence in Spark

Spark has one semi-idempotent way to save data

```
dataset.write.mode("overwrite").save(…)
```

After standardizing on idempotency, Swoop now auto-recovers from 98+% of failures

(see http://bit.ly/spark-idempotency talk)

idempotence is the second most valuable property of your Spark jobs

(after correctness)

Problem: root cause analysis

While ingesting 100M pieces of data, your code throws 1,000 exceptions of 5 different types.

How will you find & fix all the problems?

Exception handling for jobs

- Spark's default: stop after first exception
- Catch & log exceptions
- Use structured logging service

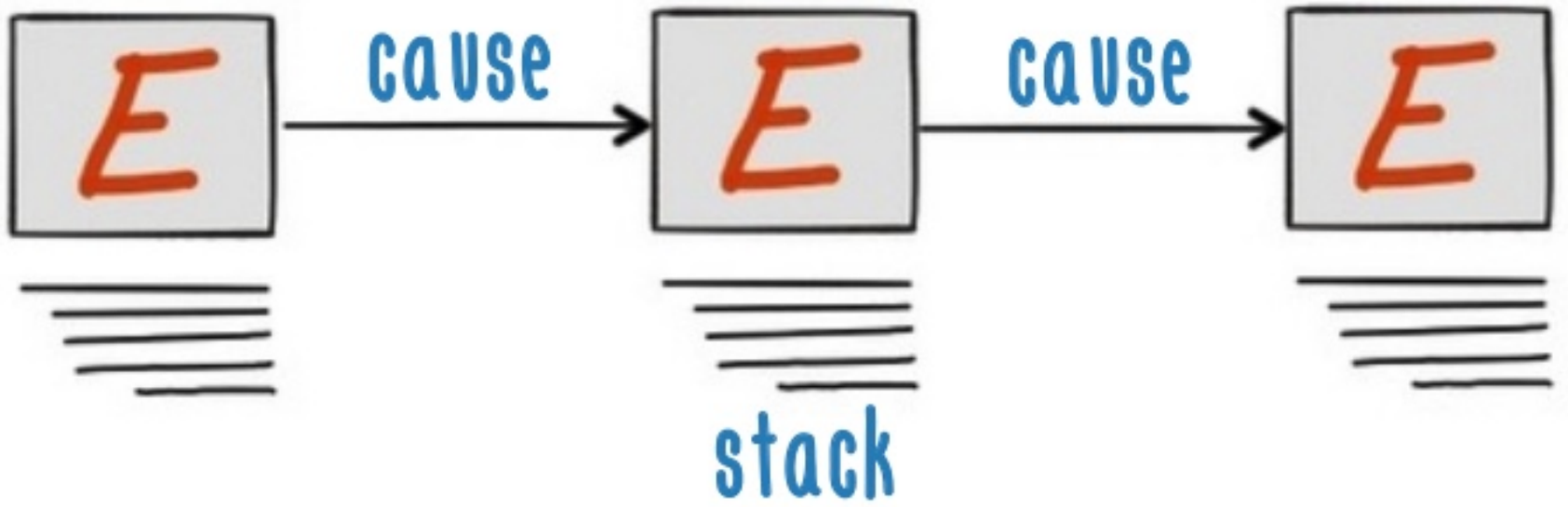
If I had a magic wand…

- Errors part of job output

Hmm, we'll have to separate data & errors…

If I had a magic wand…

- Errors part of job output
- Detailed exception info



If I had a magic wand…

- Errors part of job output
- Detailed exception info
- Flexible error summaries

category_type	id_message	cnt	messages
E	number out of range	5	▸ ["E01001: input outside [0, 100]"]
E	null	1	▸ ["java.lang.ArrayIndexOutOfBoundsException: 100"]

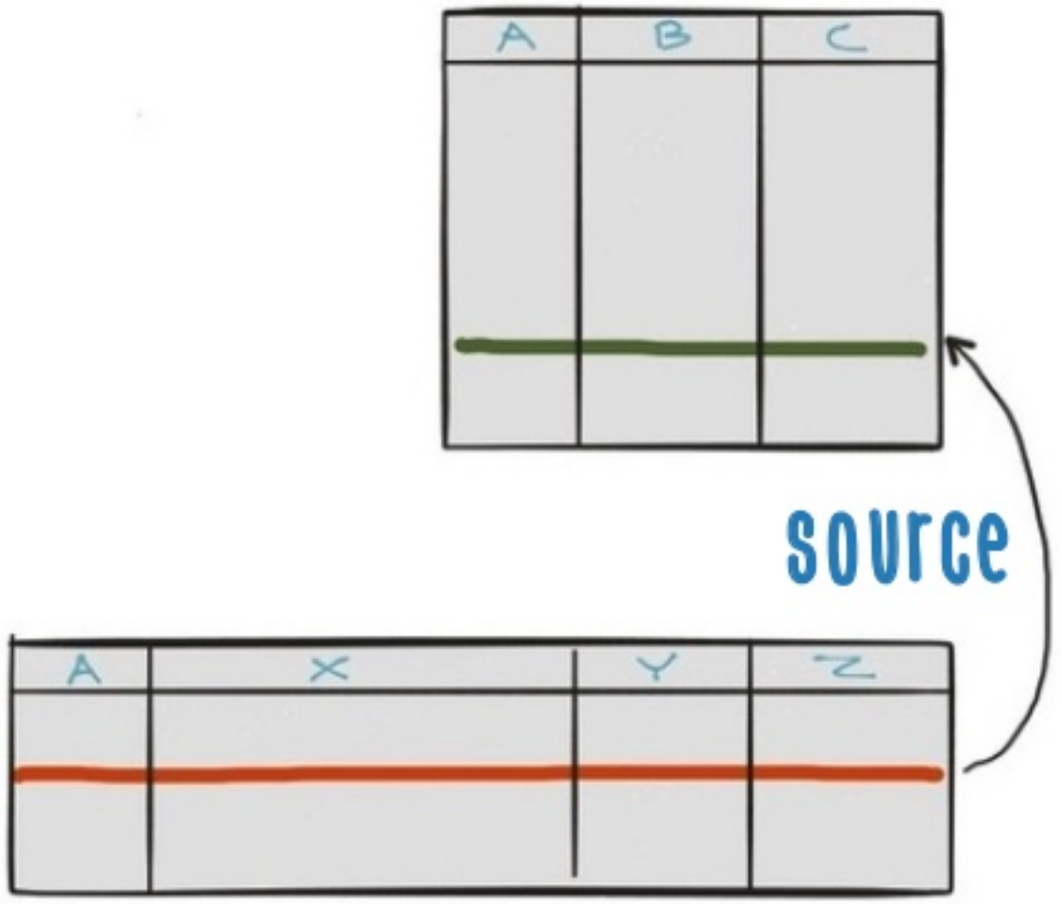
If I had a magic wand…

- Errors part of job output
- Detailed exception info
- Flexible error summaries
- Drill into errors

```
dataset.errorDetails()
dataset.unknownErrorDetails()
```

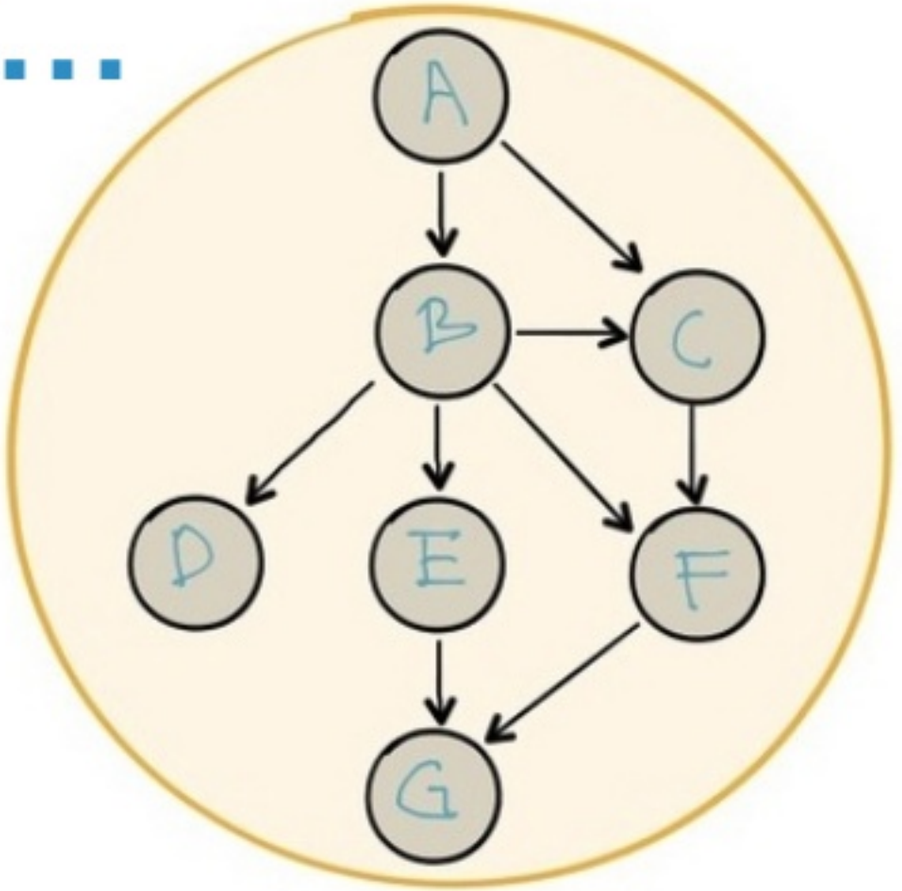
If I had a magic wand…

- Errors part of job output
- Detailed exception info
- Flexible error summaries
- Drill into errors
- Errors tied to source data



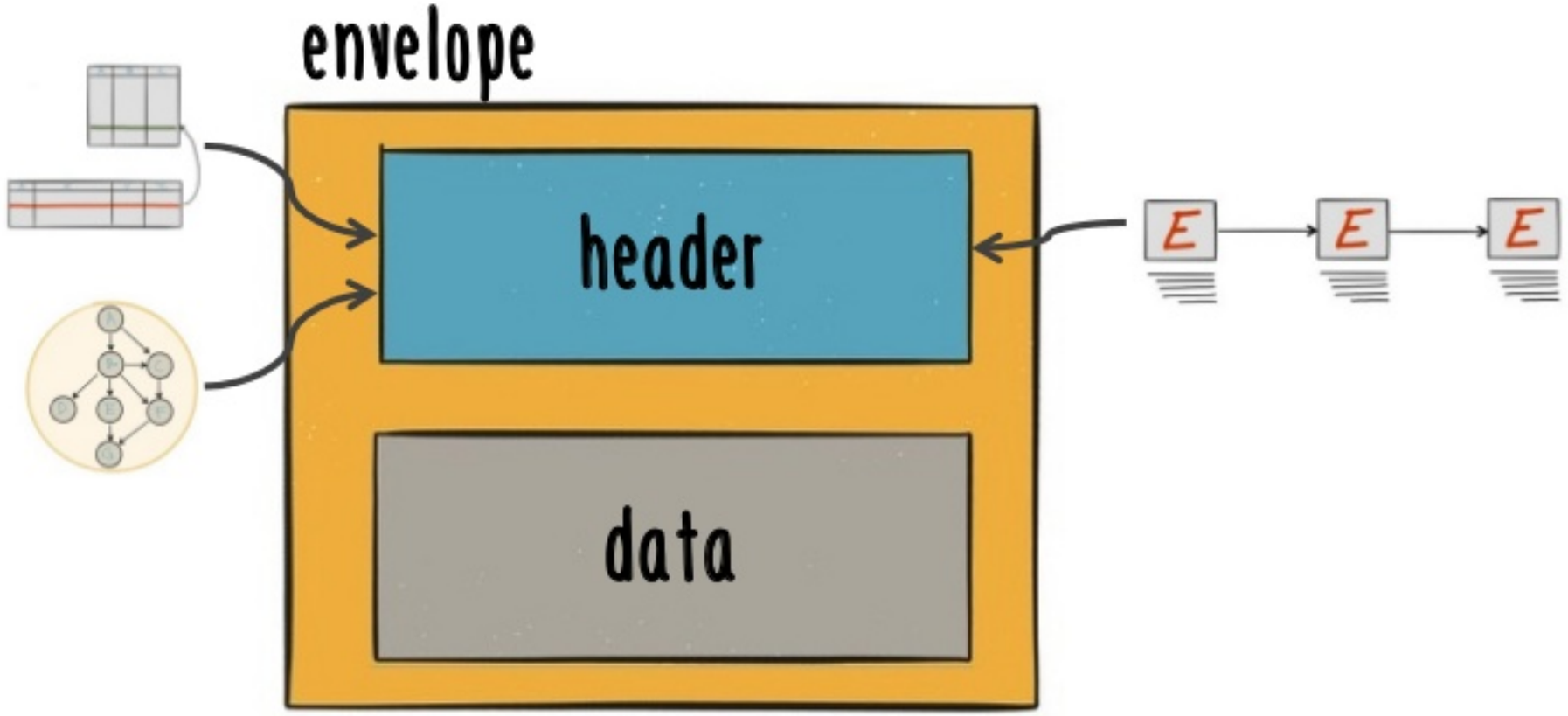
If I had a magic wand…

- Errors part of job output
- Detailed exception info
- Flexible error summaries
- Drill into errors
- Errors tied to source data
- Errors tied to job flight



job flight

Smart convention: envelopes



Spark Records

GitHub

Documentation

bulletproof Spark jobs

View on GitHub

Spark Records

```
import com.swoop.spark.records._

case class MyData(/* whatever your data is */)

case class MyDataRecord(
  features: Int,                          // fast categorization
  data:     Option[MyData] = None,        // your data
  source:   Option[MyInput] = None,       // provenance tracking
  flight:   Option[String] = None,        // related jobs
  issues:   Option[Seq[Issue]] = None     // row-level “log file”
) extends Record[MyData, MyInput]
```

Record builders

```
case class Builder(input: MyInput, jc: JobContext)
  extends RecordBuilder[…](input, jc) {

  def buildData = { /* your existing code here */ }

  def dataRecord(data: MyData, issues: Seq[Issue]) =
    MyDataRecord(…)

  def errorRecord(issues: Seq[Issue]) =
    MyDataRecord(…)
}
```

Transparent data pattern

```
// Get the data, skip all error records
records.recordData
  .createOrReplaceGlobalView("my_data")
```

any data persistence. near-zero overhead.

Demo: root cause analysis

Smart conventions & 15 lines of code

- Bulletproof job execution
- Automatic row-level structured logging
- Automatic metric collection & data quality checks
- Identifiable errors & lightning-fast RCA
- Fully customizable yet fully transparent
- Near-zero overhead

any complex data production using custom code would benefit from the Spark Records pattern

(use the schema pattern if you can't use the code)

Swoop's Spark Magic Toolkit

- Spark Records (bundled with Spark Metrics)
- Idempotent I/O to files systems & databases
  - Resilient Partitioned Tables
  - Semi-permanent tables
- Parallel DAG execution in Spark notebooks
- Smart Data Warehouse

Swoop's magic has given us

5-10x faster exploration & ML

15x more reliable job execution

10-100x faster root cause analysis

Get Swoop's Magic & Share Yours

http://bit.ly/spark-records

Email spark@swoop.com for more
or tweet @simeons

Thank You.

Get Swoop's magic & share yours.

http://bit.ly/spark-records

spark@swoop.com