

# Machine Learning & Spark

MACHINE LEARNING WITH APACHE SPARK



**Andrew Collier**  
Data Scientist, Exegetic Analytics

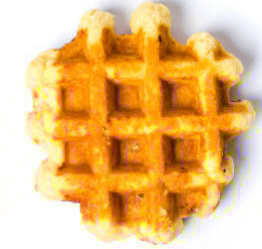
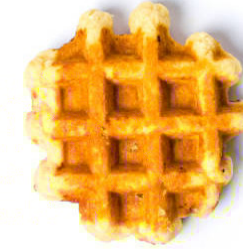
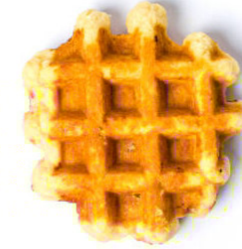
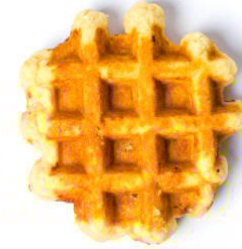
# Building the perfect waffle (an analogy)



## Archetype Waffle

Find waffle recipe. Give explicit instructions:

- 125 g flour
- 1 t baking powder
- 1 egg
- 225 ml milk
- 1 T melted butter



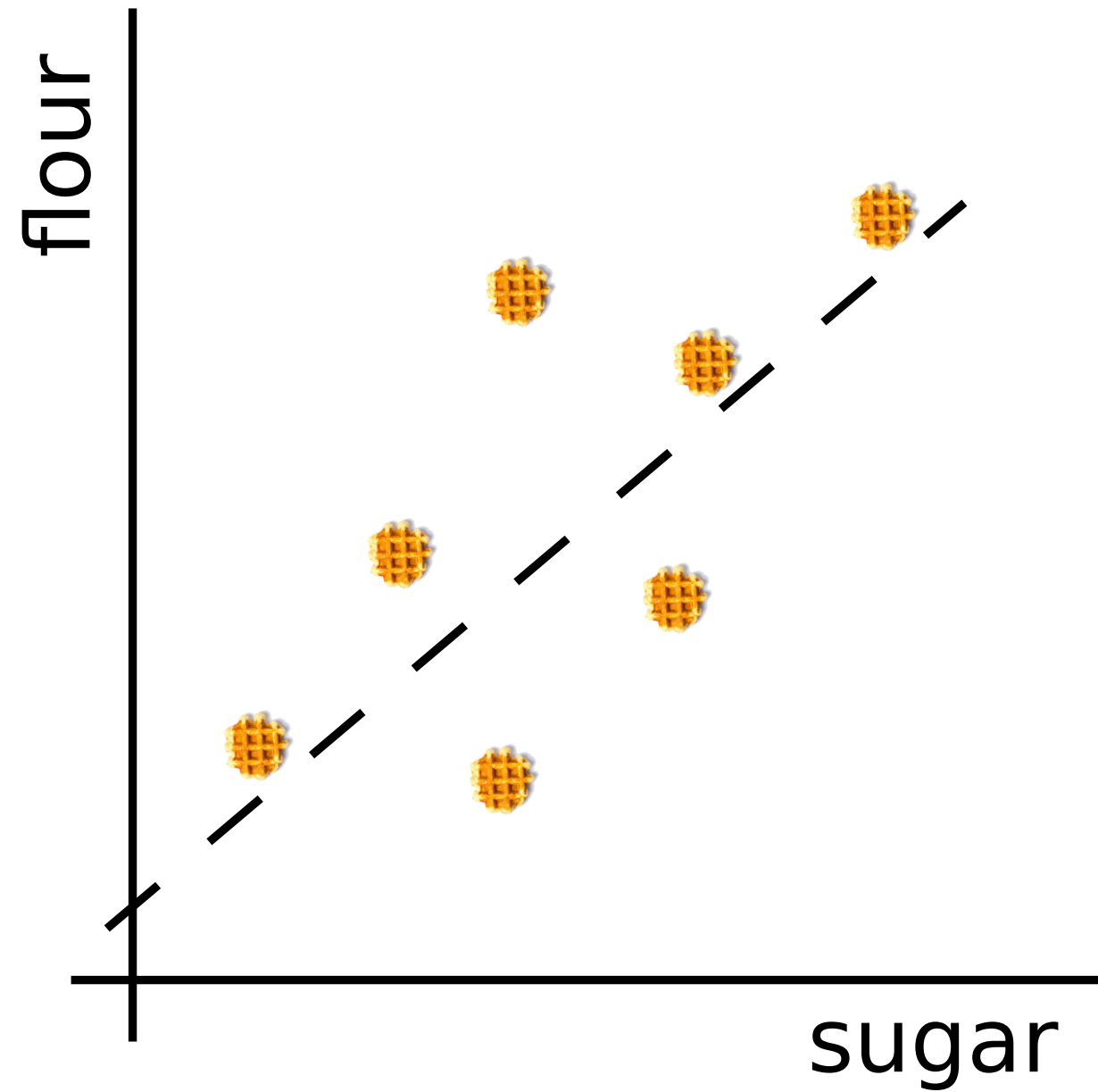
Find many waffle recipes.

Learn the perfect recipe:

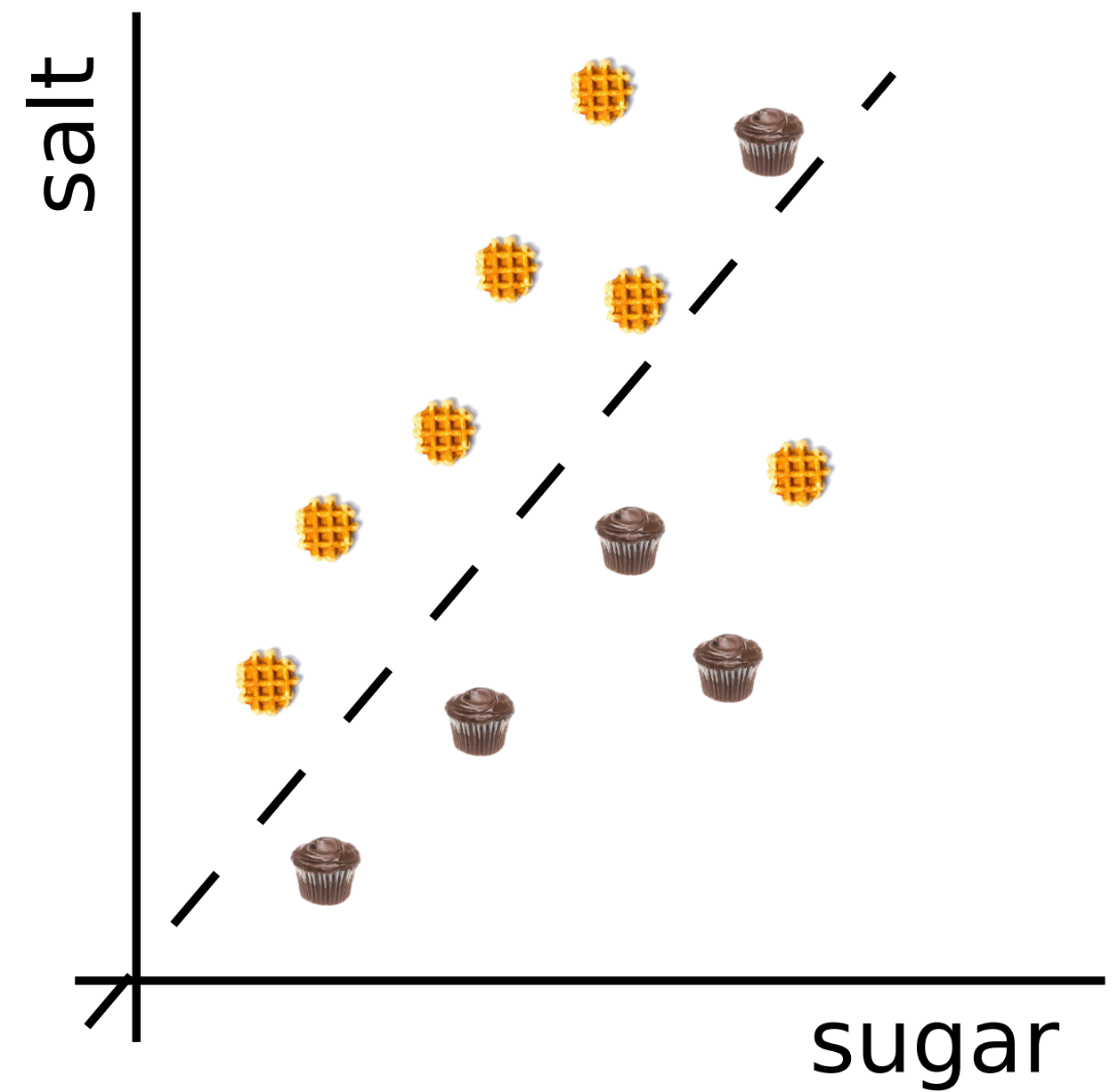
1. Look at lots of recipes.
2. What ingredients?
3. What proportions?

Computer generates its own instructions.

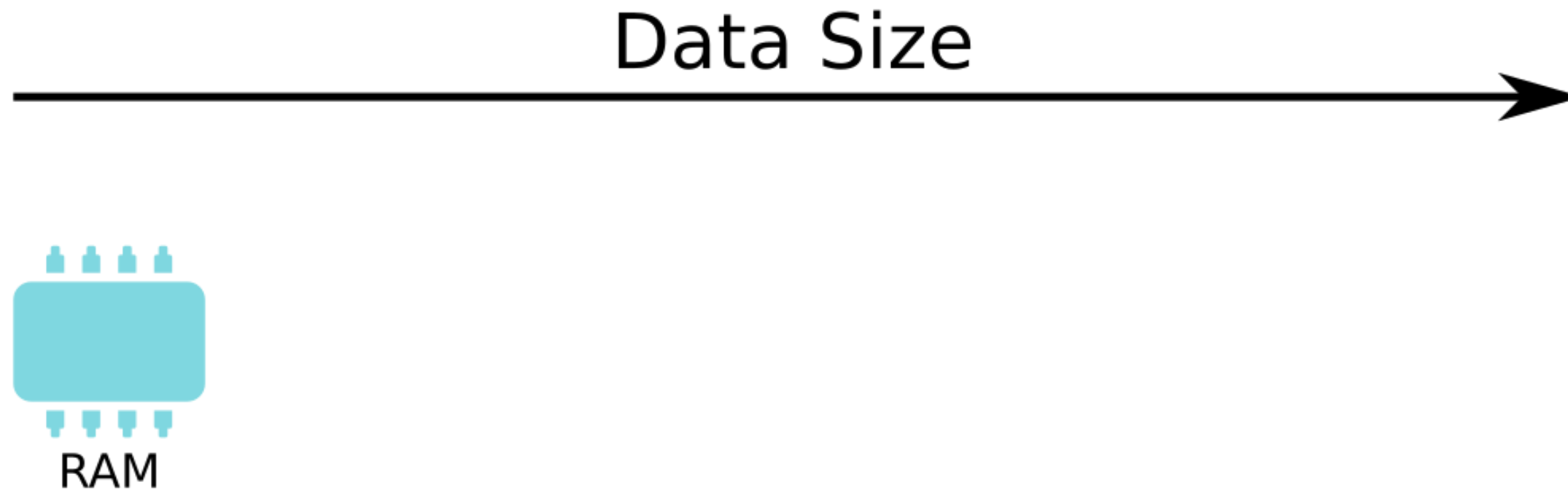
# Regression



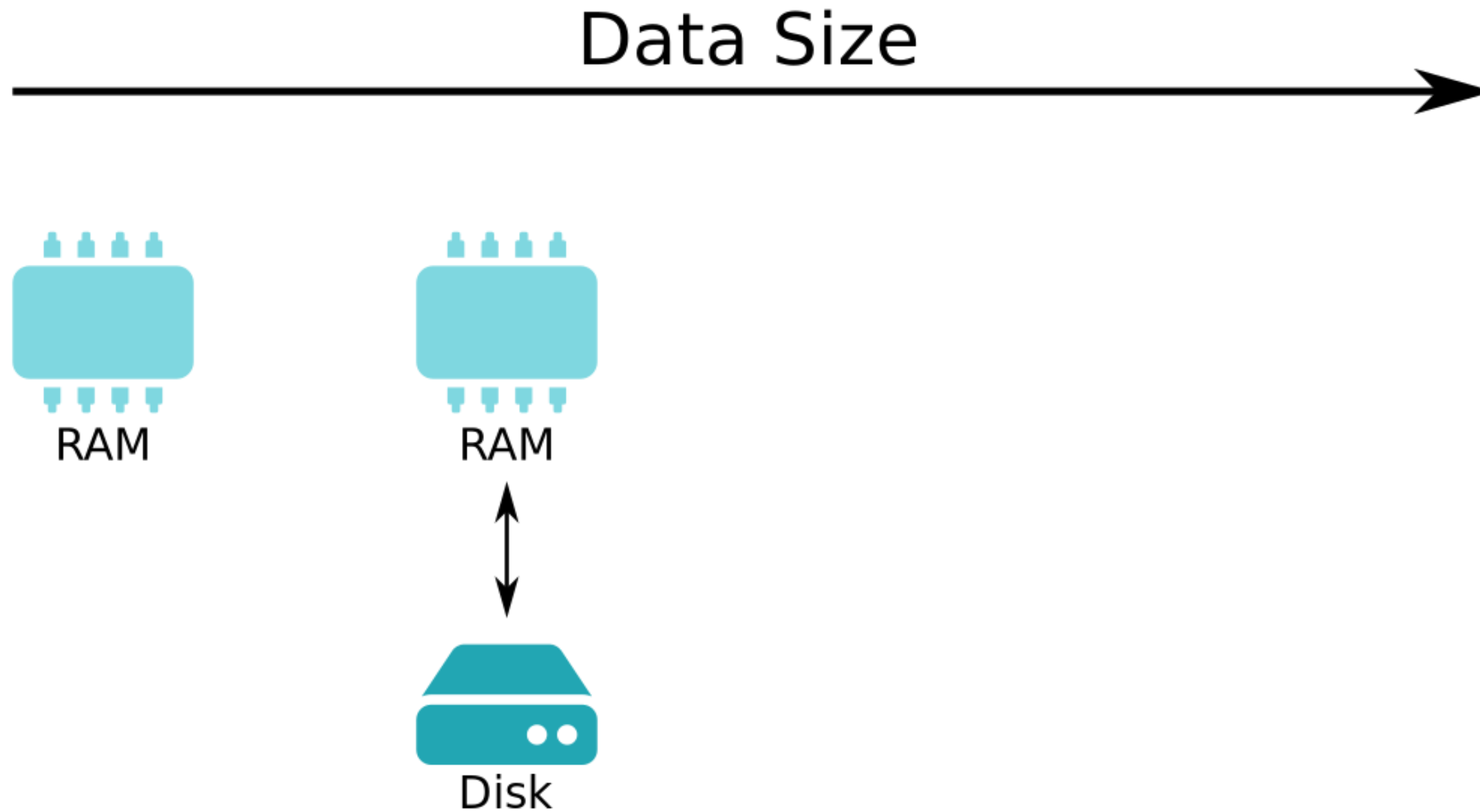
# Classification



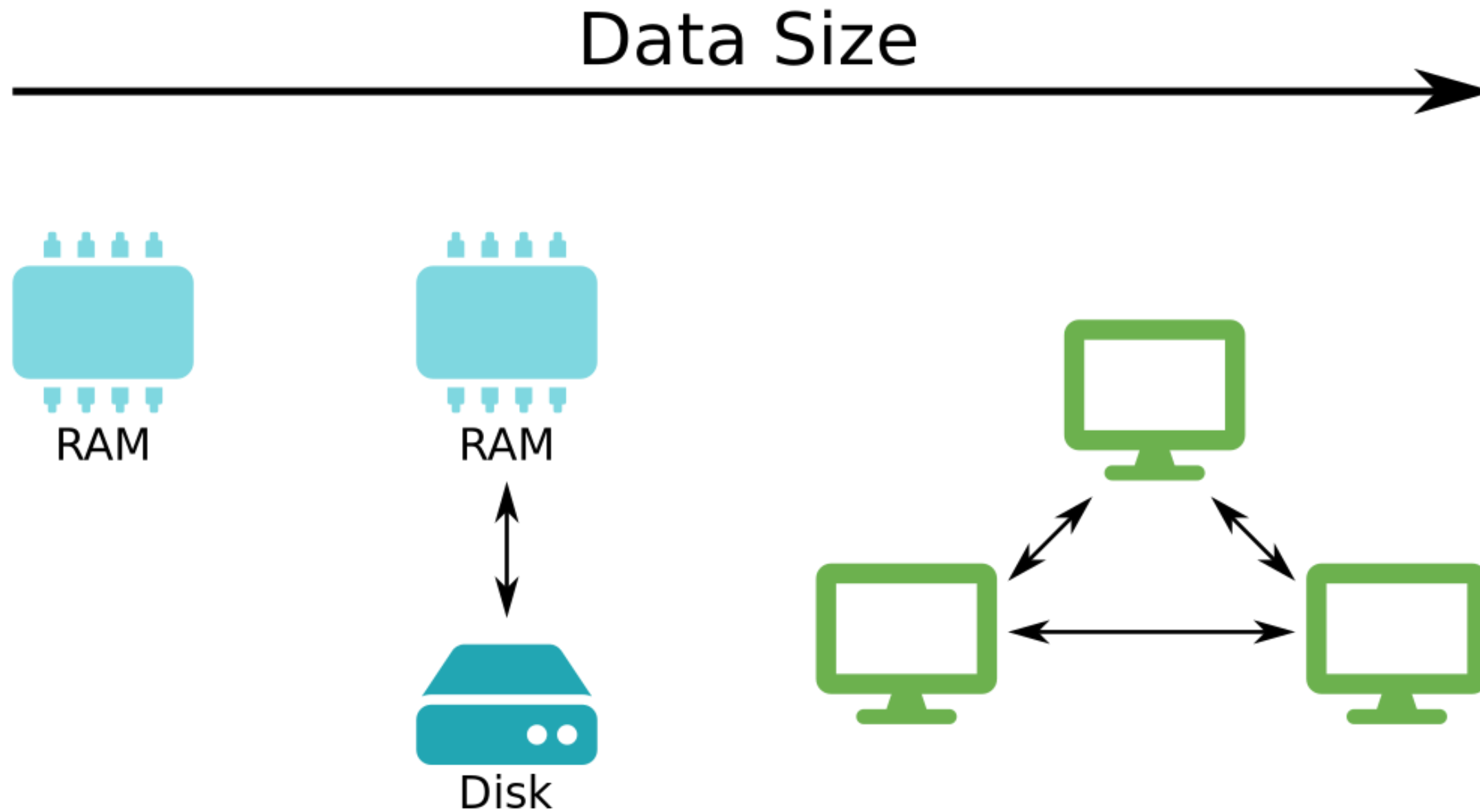
# Data in RAM



# Data exceeds RAM



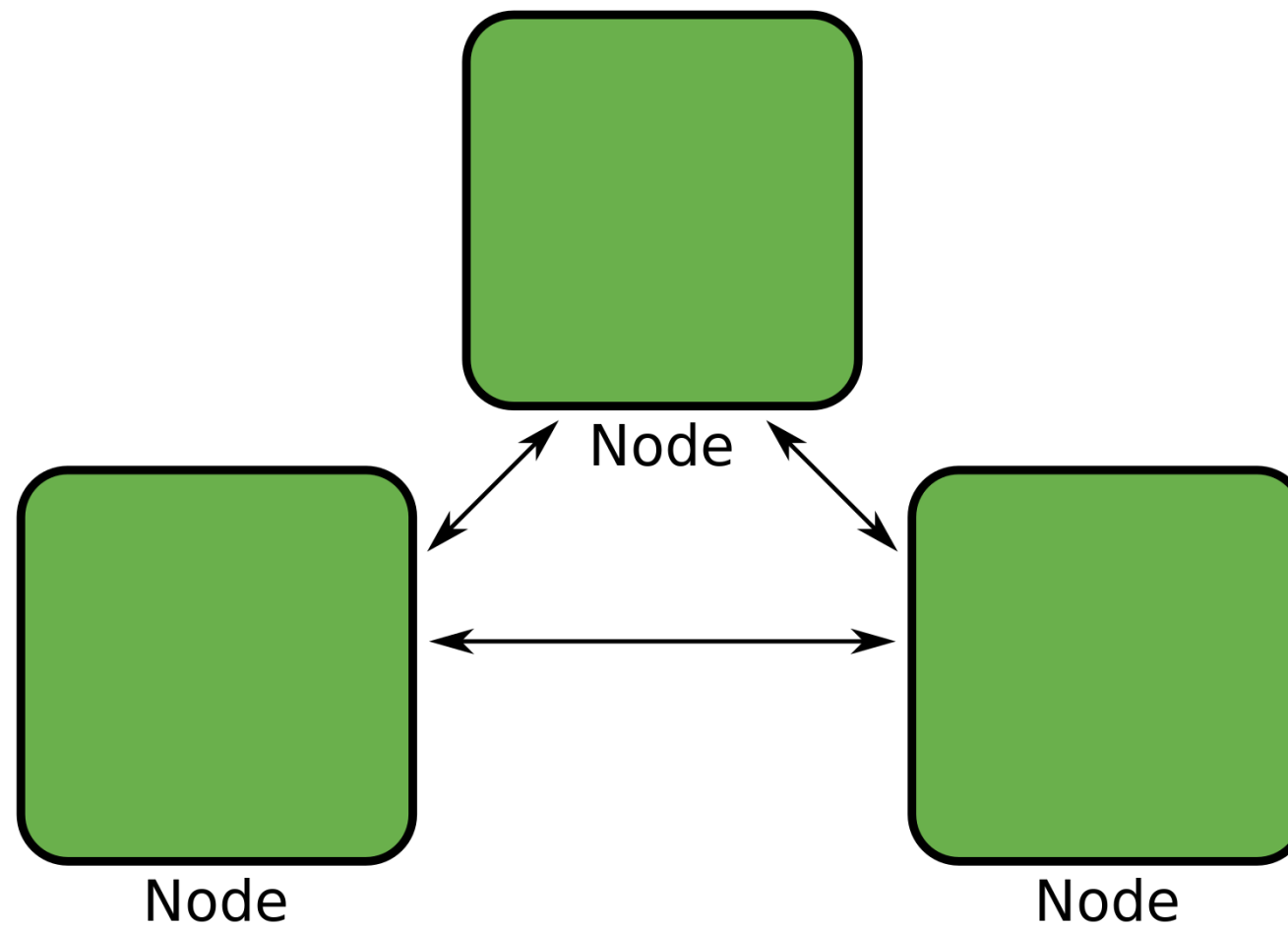
# Data distributed across a cluster



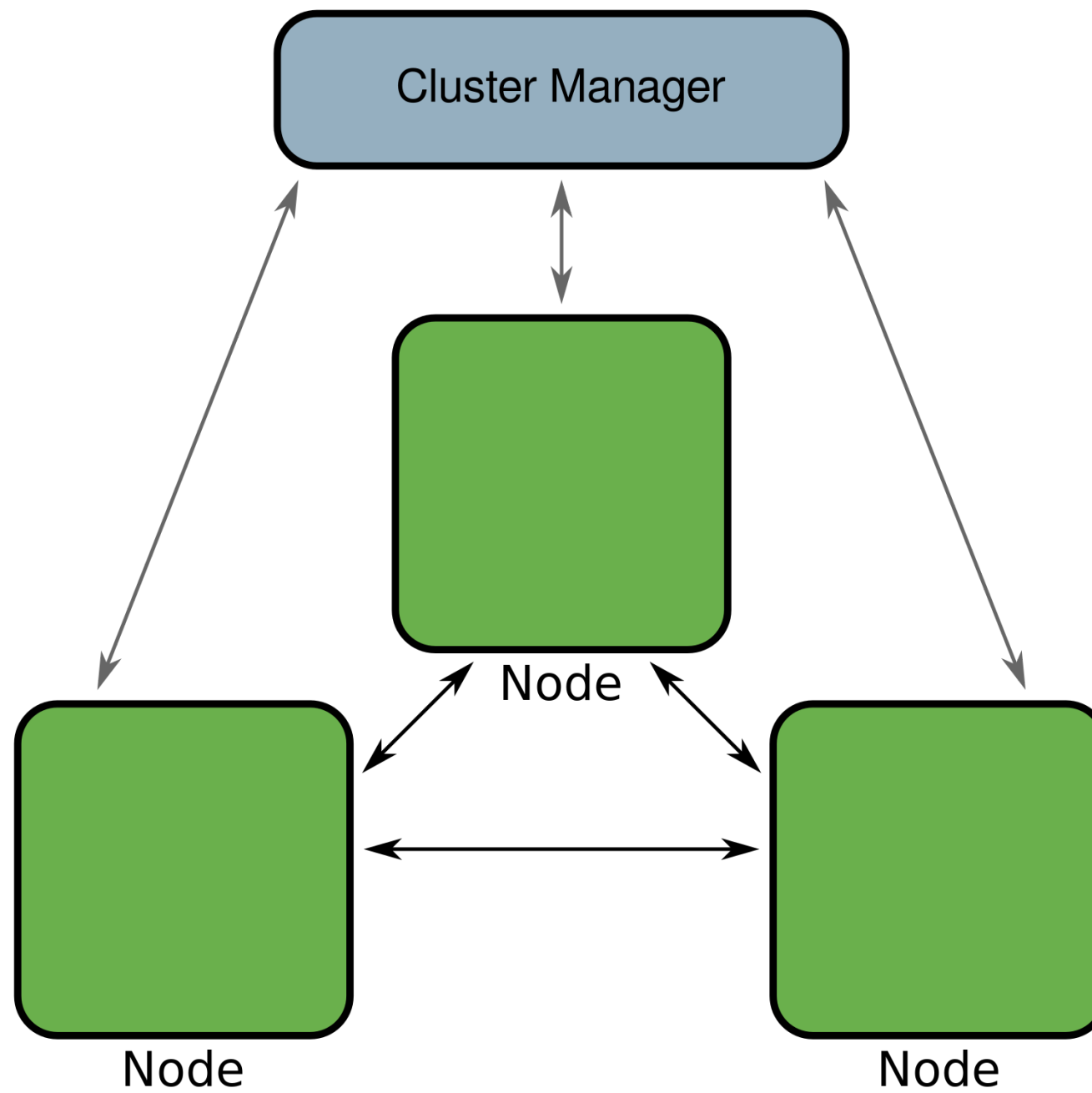
# What is Spark?

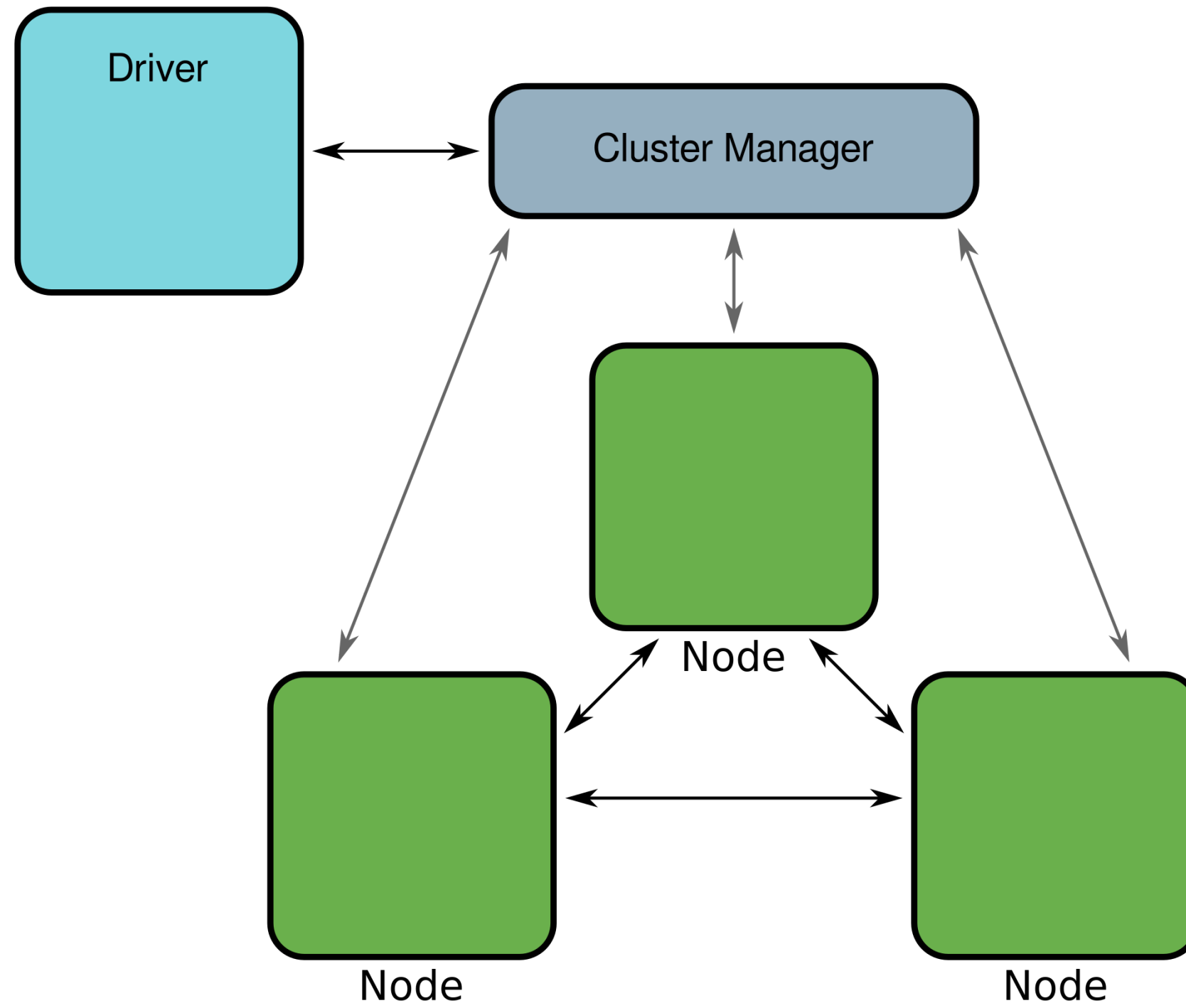


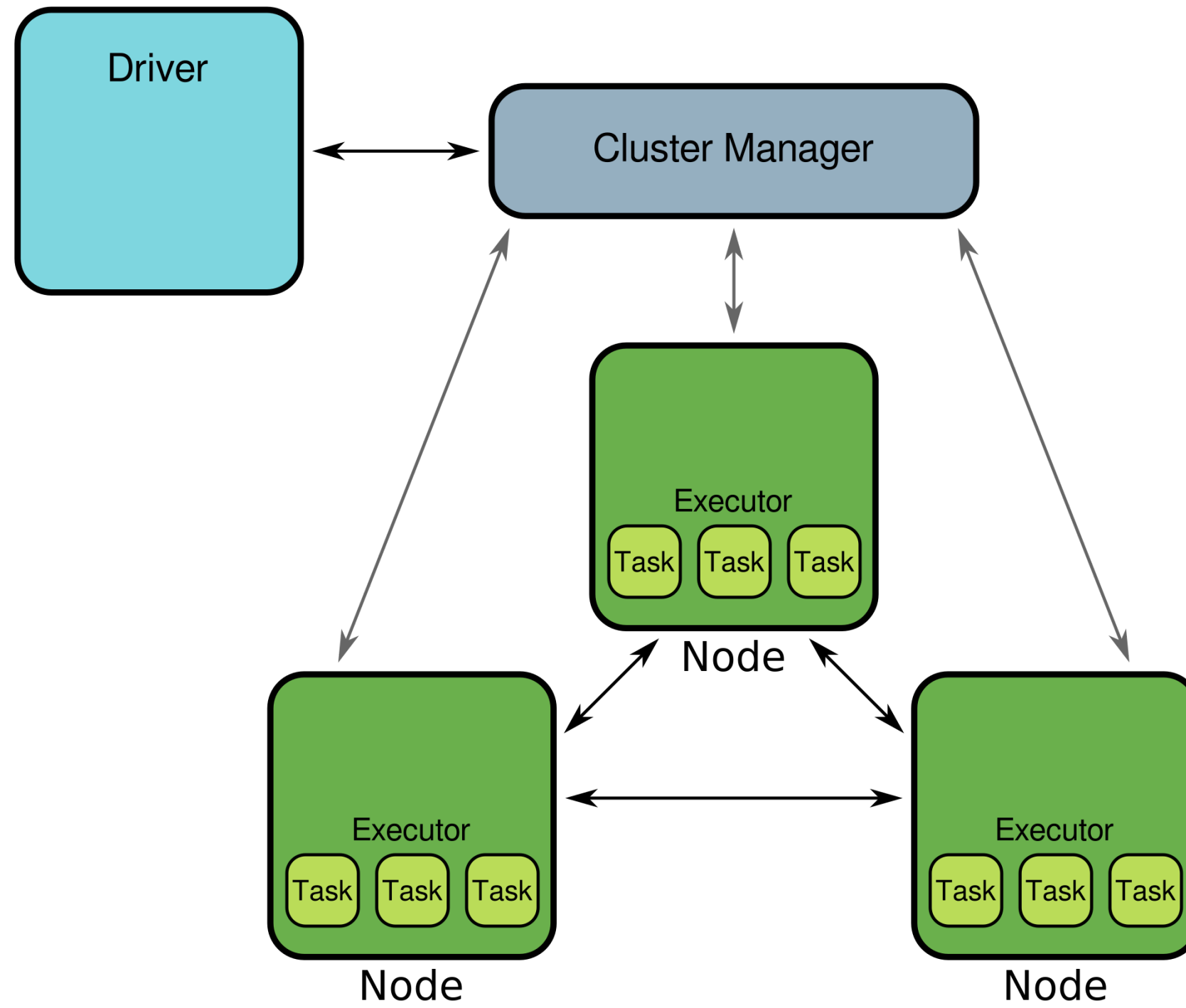
- Compute across a distributed **cluster**.
- Data processed in memory.
- Well documented high-level **API**.











# Onward!

MACHINE LEARNING WITH APACHE SPARK

# Connecting to Spark

MACHINE LEARNING WITH APACHE SPARK



**Andrew Collier**

Data Scientist, Exegetic Analytics

# Interacting with Spark

Java



Scala



Python



R



Languages for interacting with Spark.

- Java — low-level, compiled
- Scala, Python and R — high-level with interactive REPL

# Importing pyspark

From Python import the `pyspark` module.

```
import pyspark
```

Check version.

```
pyspark.__version__
```

```
'2.4.1'
```

# Sub-modules

In addition to `pyspark` there are

- Structured Data — `pyspark.sql`
- Streaming Data — `pyspark.streaming`
- Machine Learning — `pyspark.mllib` (deprecated) and `pyspark.ml`



# Spark URL

Remote Cluster using Spark URL — `spark://<IP address | DNS name>:<port>`

*Example:*

- `spark://13.59.151.161:7077`

## Local Cluster

*Examples:*

- `local` — only 1 core;
- `local[4]` — 4 cores; or
- `local[*]` — all available cores.

# Creating a SparkSession

```
from pyspark.sql import SparkSession
```

Create a local cluster using a `SparkSession` builder.

```
spark = SparkSession.builder \
    .master('local[*]') \
    .appName('first_spark_application') \
    .getOrCreate()
```

*Interact with Spark...*

```
# Close connection to Spark
>>> spark.stop()
```

# Let's connect to Spark!

MACHINE LEARNING WITH APACHE SPARK

# Loading Data

MACHINE LEARNING WITH APACHE SPARK



**Andrew Collier**

Data Scientist, Exegetic Analytics

# DataFrames: A refresher

`DataFrame` for tabular data.

123	abc	123	abc
123	abc	123	abc
123	abc	123	abc
123	abc	123	abc
123	abc	123	abc

Selected methods:

- `count()`
- `show()`
- `printSchema()`

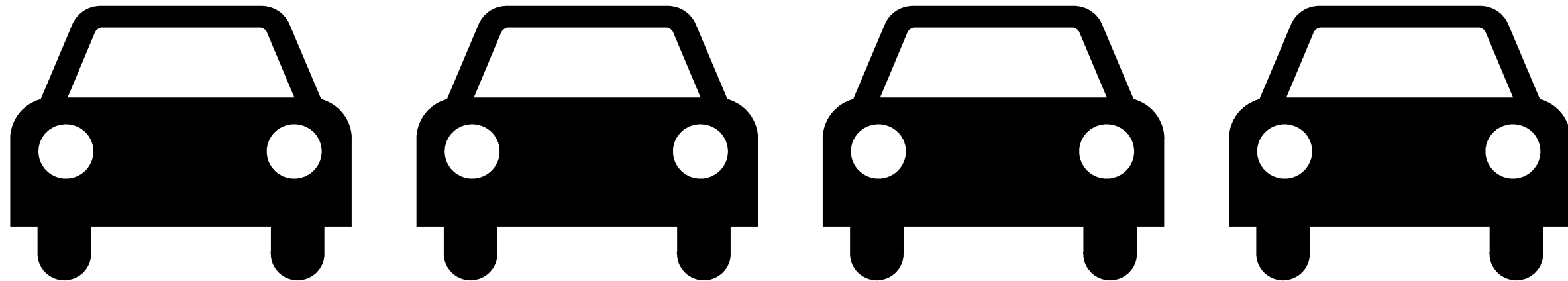
Selected attributes:

- `dtypes`

# CSV data for cars

The first few lines from the 'cars.csv' file.

```
mfr,mod,org,type,cyl,size,weight,len,rpm,cons  
Mazda,RX-7,non-USA,Sporty,NA,1.3,2895,169,6500,9.41  
Nissan,Maxima,non-USA,Midsize,6,3,3200,188,5200,9.05  
Chevrolet,Cavalier,USA,Compact,4,2.2,2490,182,5200,6.53  
Subaru,Legacy,non-USA,Compact,4,2.2,3085,179,5600,7.84  
Ford,Escort,USA,Small,4,1.8,2530,171,6500,7.84
```



# Reading data from CSV

The `.csv()` method reads a CSV file and returns a `DataFrame`.

```
cars = spark.read.csv('cars.csv', header=True)
```

Optional arguments:

- `header` — is first row a header? (default: `False`)
- `sep` — field separator (default: a comma `,`)
- `schema` — explicit column data types
- `inferSchema` — deduce column data types from data?
- `nullValue` — placeholder for missing data

# Peek at the data

The first five records from the `DataFrame` .

```
cars.show(5)
```

```
+-----+-----+-----+-----+--+-----+-----+-----+
|      mfr|      mod|      org|  type|cyl|size|weight|len| rpm|cons|
+-----+-----+-----+-----+--+-----+-----+-----+
|    Mazda|    RX-7|non-USA| Sporty| NA| 1.3| 2895|169|6500|9.41|
|   Nissan|  Maxima|non-USA| Midsize|  6|  3| 3200|188|5200|9.05|
|Chevrolet|Cavalier|    USA| Compact|  4| 2.2| 2490|182|5200|6.53|
|   Subaru|  Legacy|non-USA| Compact|  4| 2.2| 3085|179|5600|7.84|
|    Ford|  Escort|    USA|   Small|  4| 1.8| 2530|171|6500|7.84|
+-----+-----+-----+-----+--+-----+-----+-----+
```



# Check column types

```
cars.printSchema()
```

```
root
|-- mfr: string (nullable = true)
|-- mod: string (nullable = true)
|-- org: string (nullable = true)
|-- type: string (nullable = true)
|-- cyl: string (nullable = true)
|-- size: string (nullable = true)
|-- weight: string (nullable = true)
|-- len: string (nullable = true)
|-- rpm: string (nullable = true)
|-- cons: string (nullable = true)
```

# Inferring column types from data

```
cars = spark.read.csv("cars.csv", header=True, inferSchema=True)
cars.dtypes
```

```
[('mfr', 'string'),
 ('mod', 'string'),
 ('org', 'string'),
 ('type', 'string'),
 ('cyl', 'string'),
 ('size', 'double'),
 ('weight', 'int'),
 ('len', 'int'),
 ('rpm', 'int'),
 ('cons', 'double')]
```

# Dealing with missing data

Handle missing data using the `nullValue` argument.

```
cars = spark.read.csv("cars.csv", header=True, inferSchema=True, nullValue='NA')
```

The `nullValue` argument is case sensitive.

# Specify column types

```
schema = StructType([
    StructField("maker", StringType()),
    StructField("model", StringType()),
    StructField("origin", StringType()),
    StructField("type", StringType()),
    StructField("cyl", IntegerType()),
    StructField("size", DoubleType()),
    StructField("weight", IntegerType()),
    StructField("length", DoubleType()),
    StructField("rpm", IntegerType()),
    StructField("consumption", DoubleType())
])

cars = spark.read.csv("cars.csv", header=True, schema=schema, nullValue='NA')
```

# Final cars data

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|maker   |model       |origin |type   |cyl  |size|weight|length|rpm  |consumption|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|Mazda    |RX-7        |non-USA|Sporty |null |1.3 |2895  |169.0 |6500|9.41      |
|Nissan   |Maxima      |non-USA|Midsize|6    |3.0 |3200  |188.0 |5200|9.05      |
|Chevrolet|Cavalier    |USA    |Compact|4    |2.2 |2490  |182.0 |5200|6.53      |
|Subaru   |Legacy      |non-USA|Compact|4    |2.2 |3085  |179.0 |5600|7.84      |
|Ford     |Escort      |USA    |Small  |4    |1.8 |2530  |171.0 |6500|7.84      |
|Mercury  |Capri       |USA    |Sporty |4    |1.6 |2450  |166.0 |5750|9.05      |
|Oldsmobile|Cutlass Ciera|USA    |Midsize|4    |2.2 |2890  |190.0 |5200|7.59      |
|Saab     |900         |non-USA|Compact|4    |2.1 |2775  |184.0 |6000|9.05      |
|Dodge    |Caravan     |USA    |Van    |6    |3.0 |3705  |175.0 |5000|11.2      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

# Let's load some data!

MACHINE LEARNING WITH APACHE SPARK