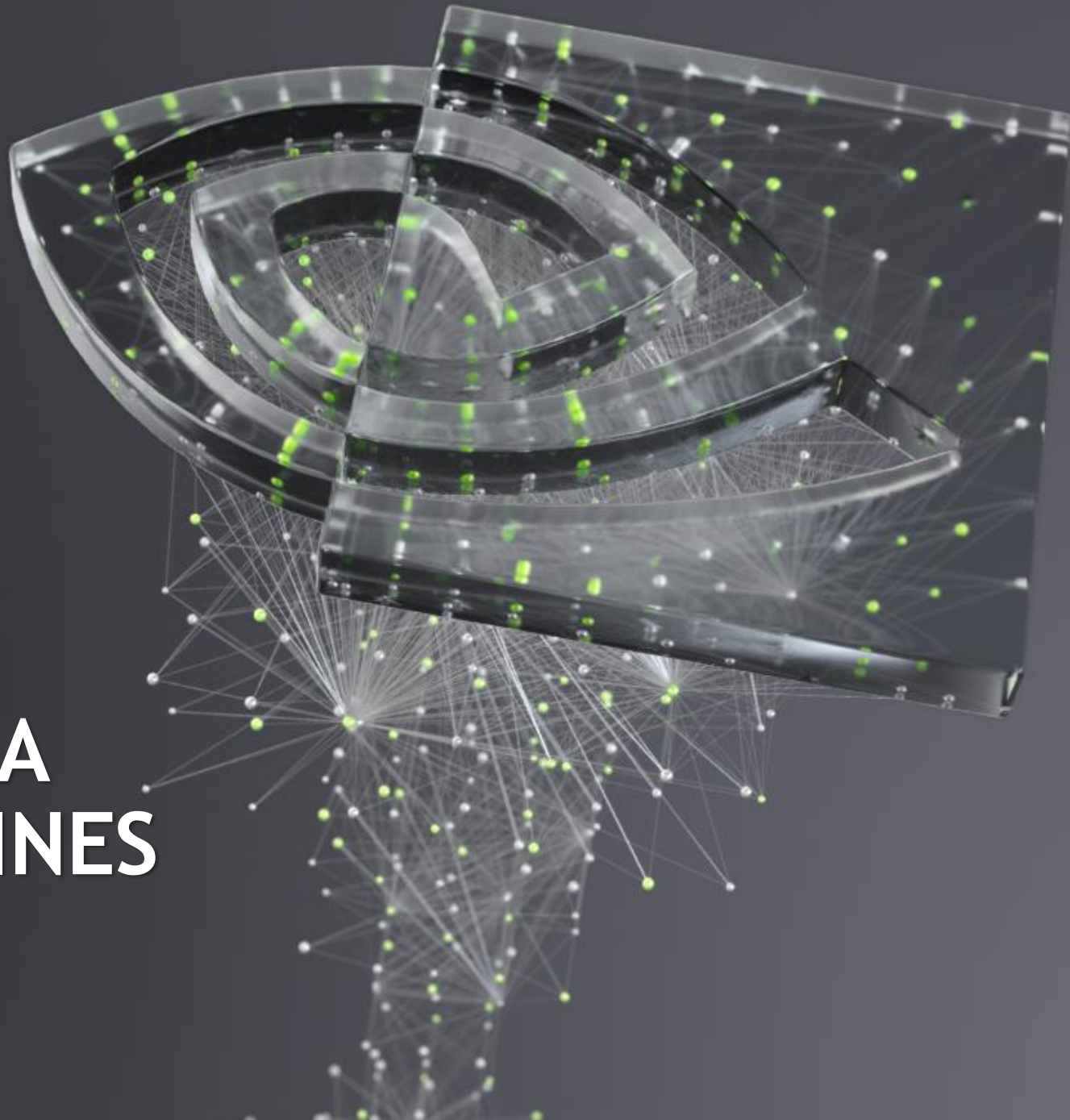


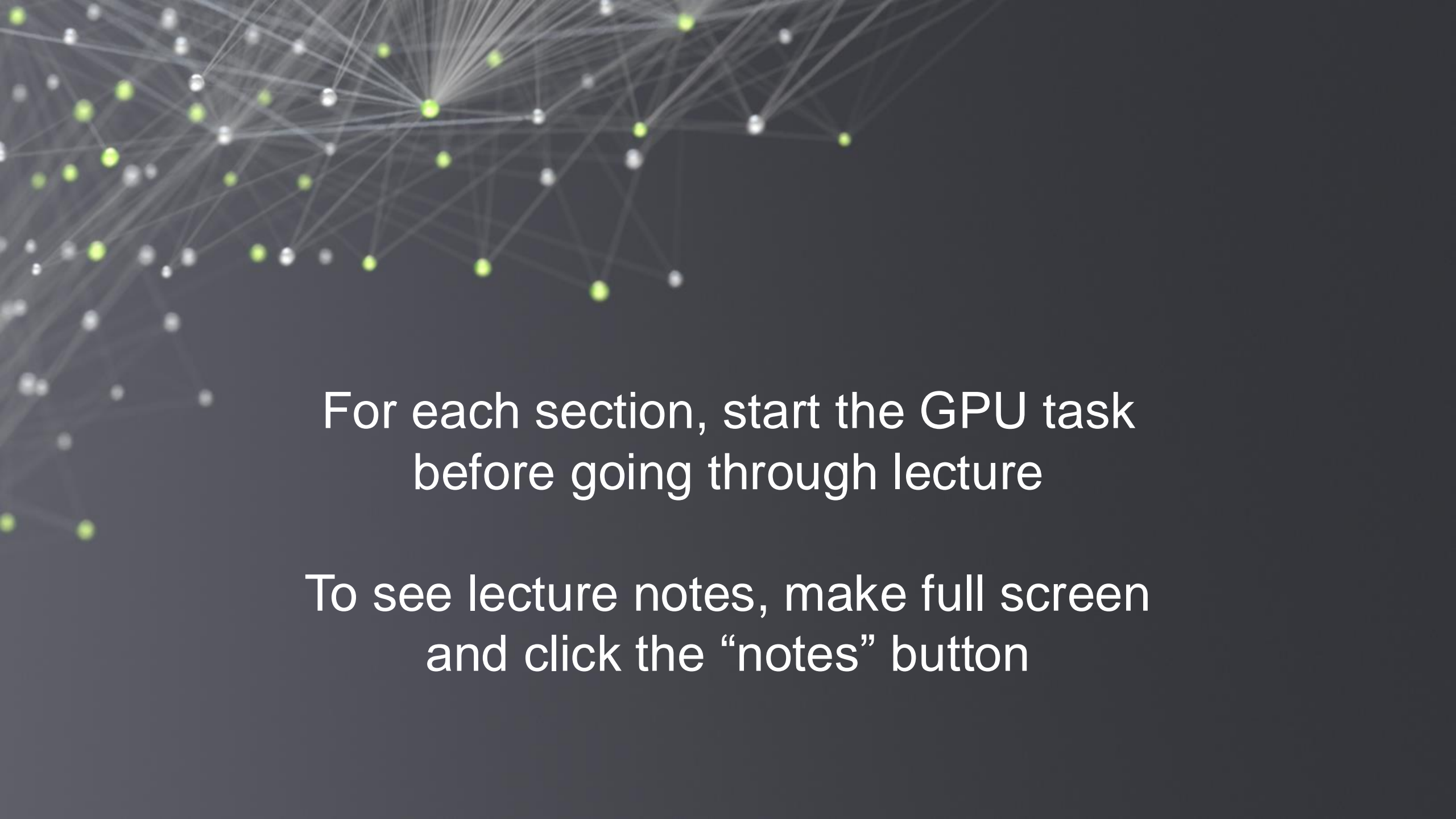


DEEP
LEARNING
INSTITUTE

ACCELERATING DATA ENGINEERING PIPELINES

Part 1: Data Storage



An abstract graphic in the top-left corner of the slide. It features a complex network of thin, light-grey lines connecting numerous small, semi-transparent nodes. Some nodes are white, while others are a bright yellow-green. The network is dense and appears to be a visualization of a graph or data structure, with lines radiating from a central point towards the edges of the frame.

For each section, start the GPU task
before going through lecture

To see lecture notes, make full screen
and click the “notes” button



WELCOME!



Main Goal:
How do we organize and process an
unexplored dataset to produce
actionable insight?

THE GOALS OF THIS COURSE

- Get used to many different data types / frameworks and how they operate on GPU vs CPU.
- Understand how DAG based frameworks can speed up ETL
- Learn how to visualize data to
 - Assess data quality
 - Allow users to make their own decisions through interactivity

AGENDA

Part 1: Data Formats

Part 2: ETL with NVTabular

Part 3: Data Visualization

AGENDA – PART I

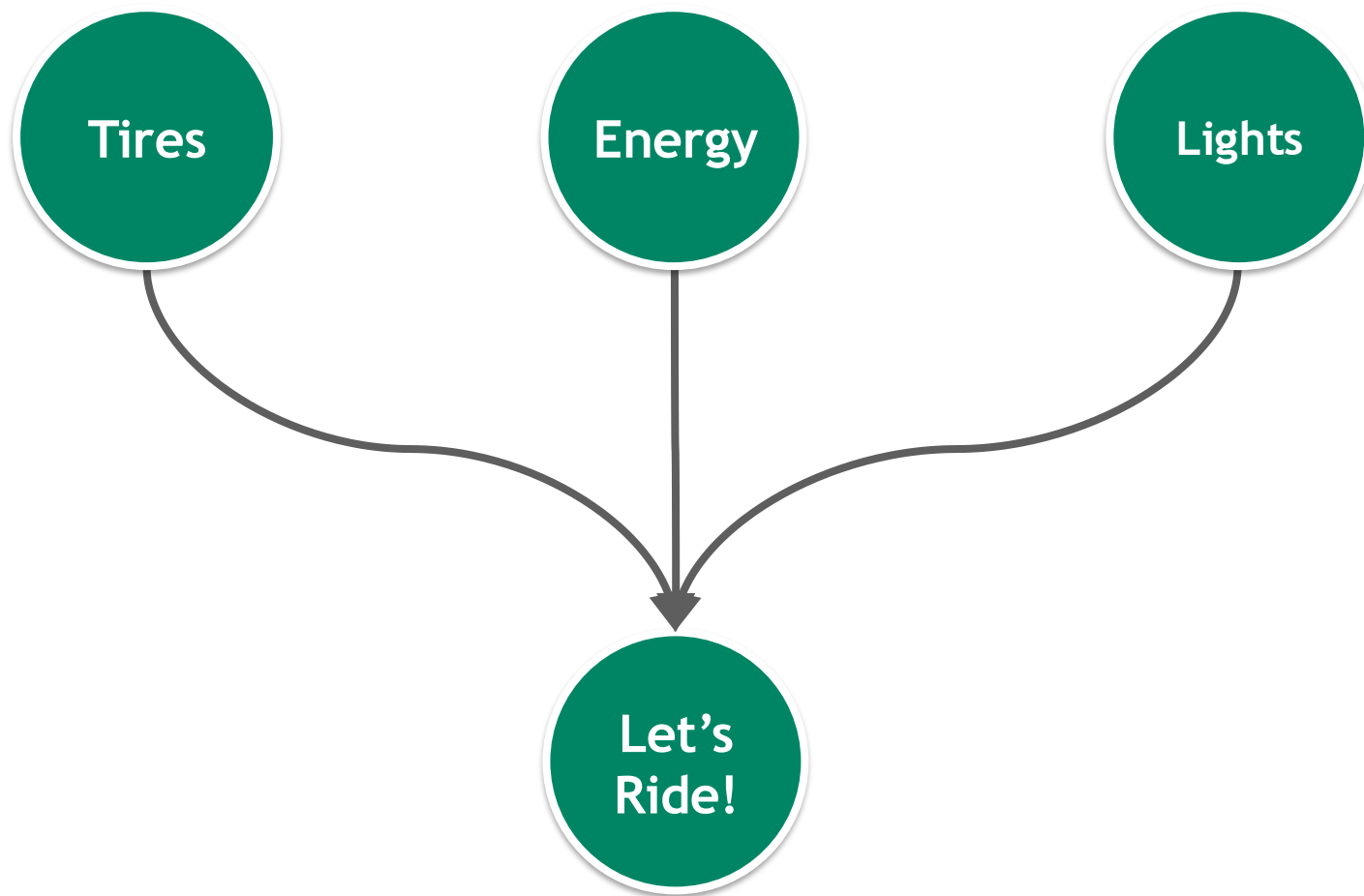
- Systems Engineering
- File Formats
- Data Frameworks
- Lab



SYSTEMS ENGINEERING

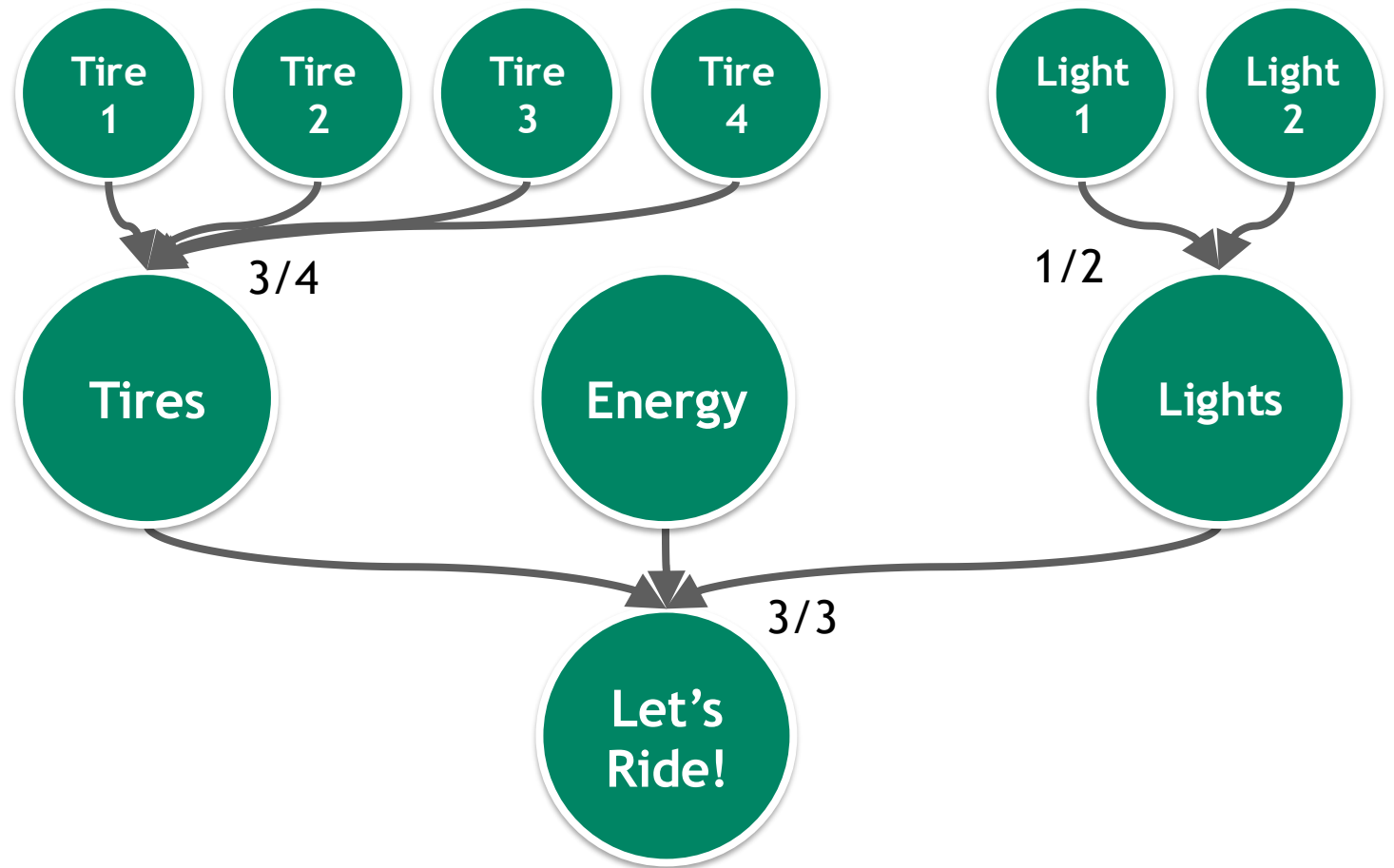
SYSTEM EXAMPLE

Modeling a Car



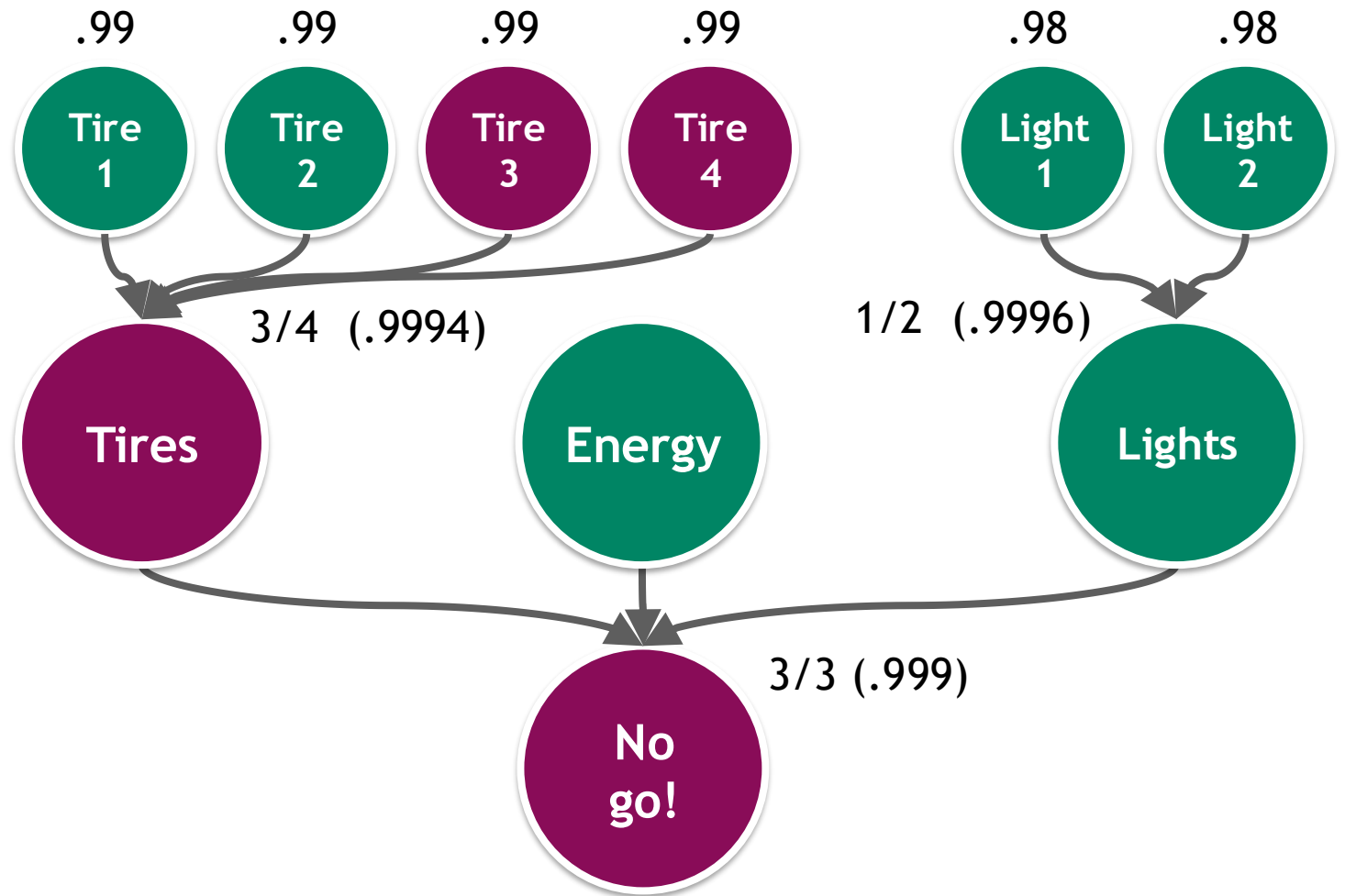
SYSTEM EXAMPLE

Modeling a Car



SYSTEM EXAMPLE

Modeling a Car



SYSTEMS BIG AND SMALL

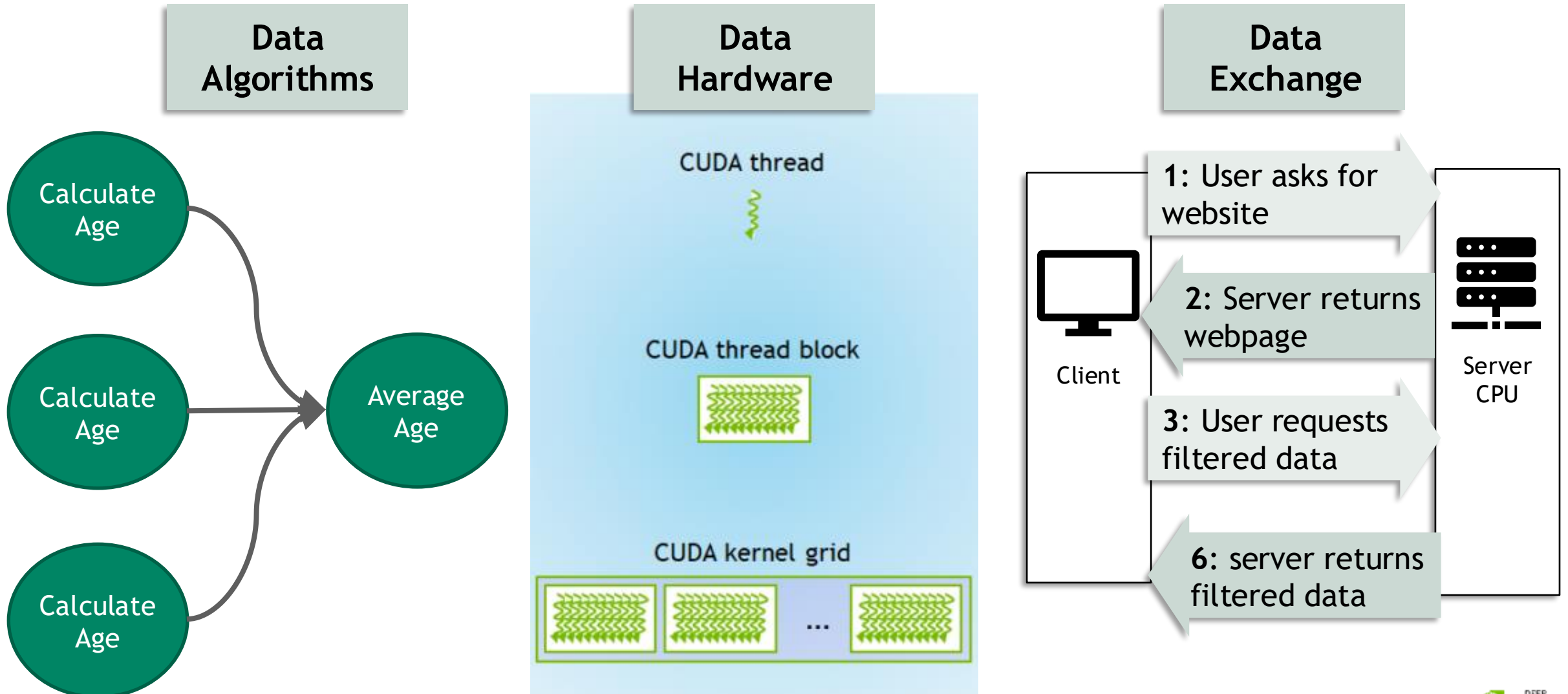
Details



Big Picture



SYSTEM ENGINEERING FOR DATA

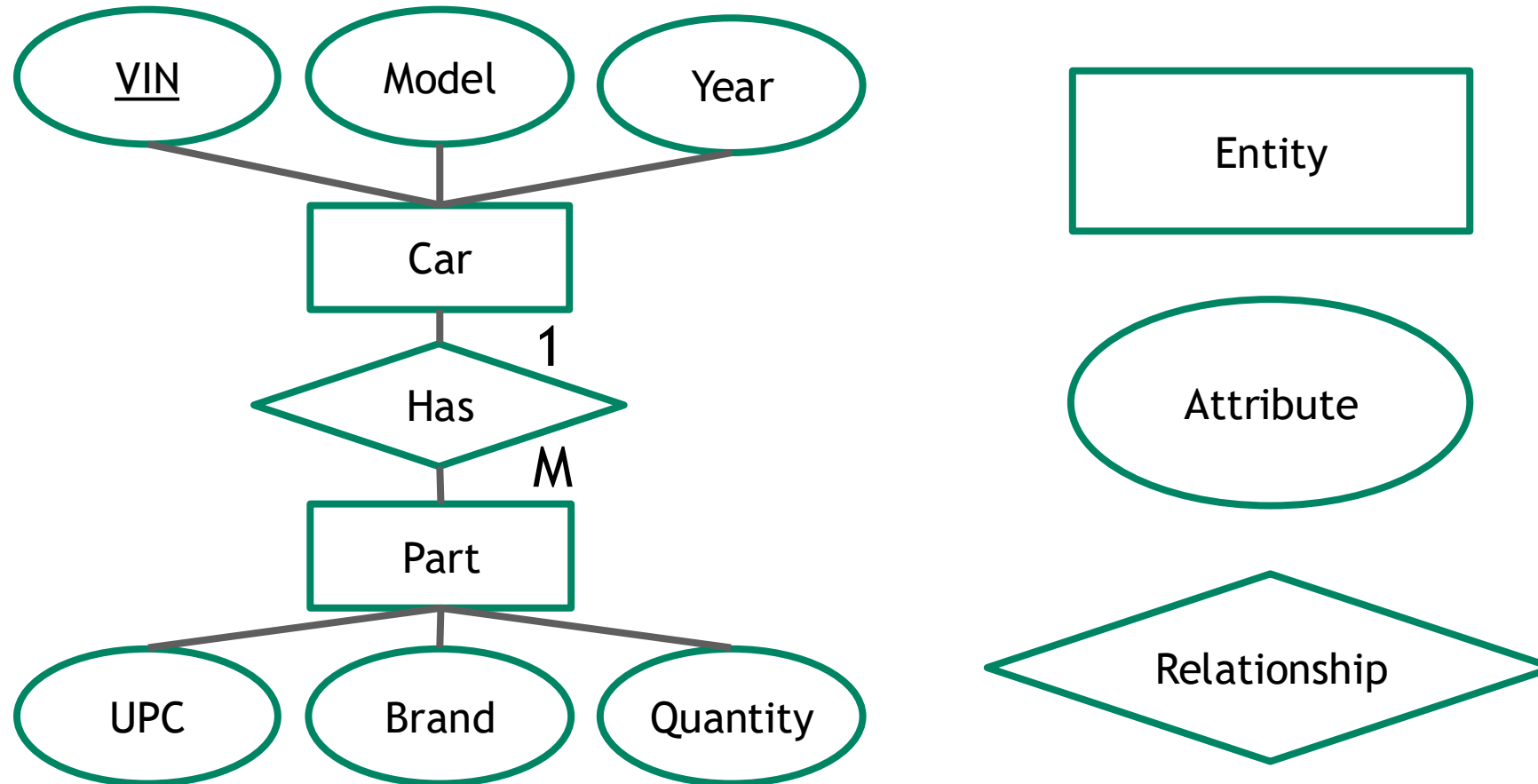




DATA AS A SYSTEM OF ALGORITHMS

DATABASE SYSTEM DESIGN

Enhanced Entity Relationship Diagram



DATABASE SYSTEM DESIGN

From Design to Practice

Cars.csv

VIN, Model, Year
1a2b3c, Sedan, 1986
4d5e6g, Convertible, 2011
7h8i9j, Sedan, 1997

Parts.csv

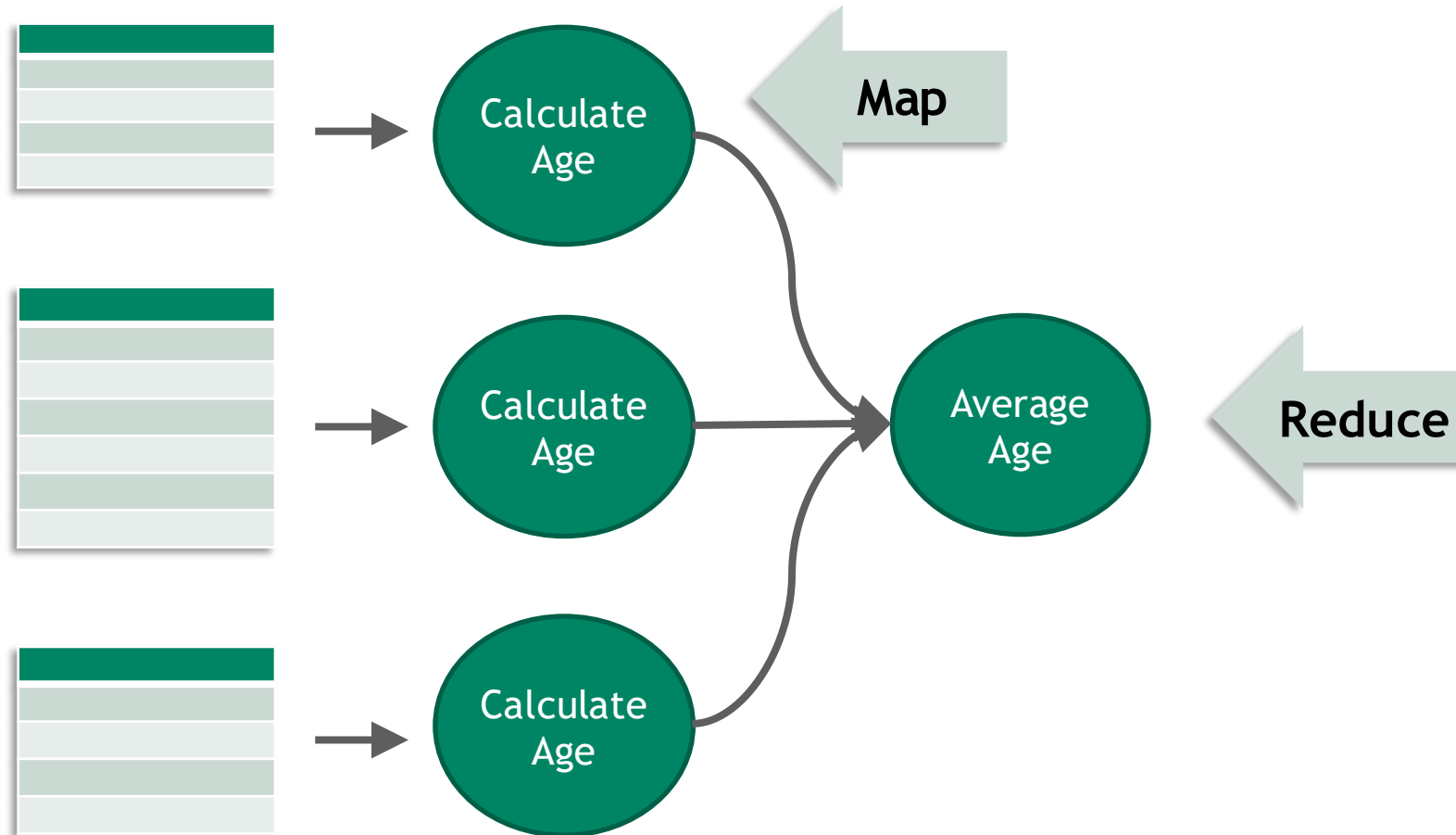
VIN, UPC, Brand, Quantity
1a2b3c, 8675309, Generic Lights, 2
1a2b3c, 8675310, Generic Tires, 4
4d5e6g, 8675309, Awesome Lights, 2
4d5e6g, 8675310, Awesome Tires, 4

Cars.json

```
{  
  {  
    "VIN": 1a3b3c,  
    "Model": "Sedan",  
    "Year": 1986,  
    "Parts": [  
      {  
        "UPC": 8675309,  
        "Brand": "Generic Lights",  
        "Quantity": 2  
      }, {  
        "UPC": 8675310,  
        "Brand": "Generic Lights",  
        "Quantity": 4  
      }  
    ]  
  },  
  ...  
}
```

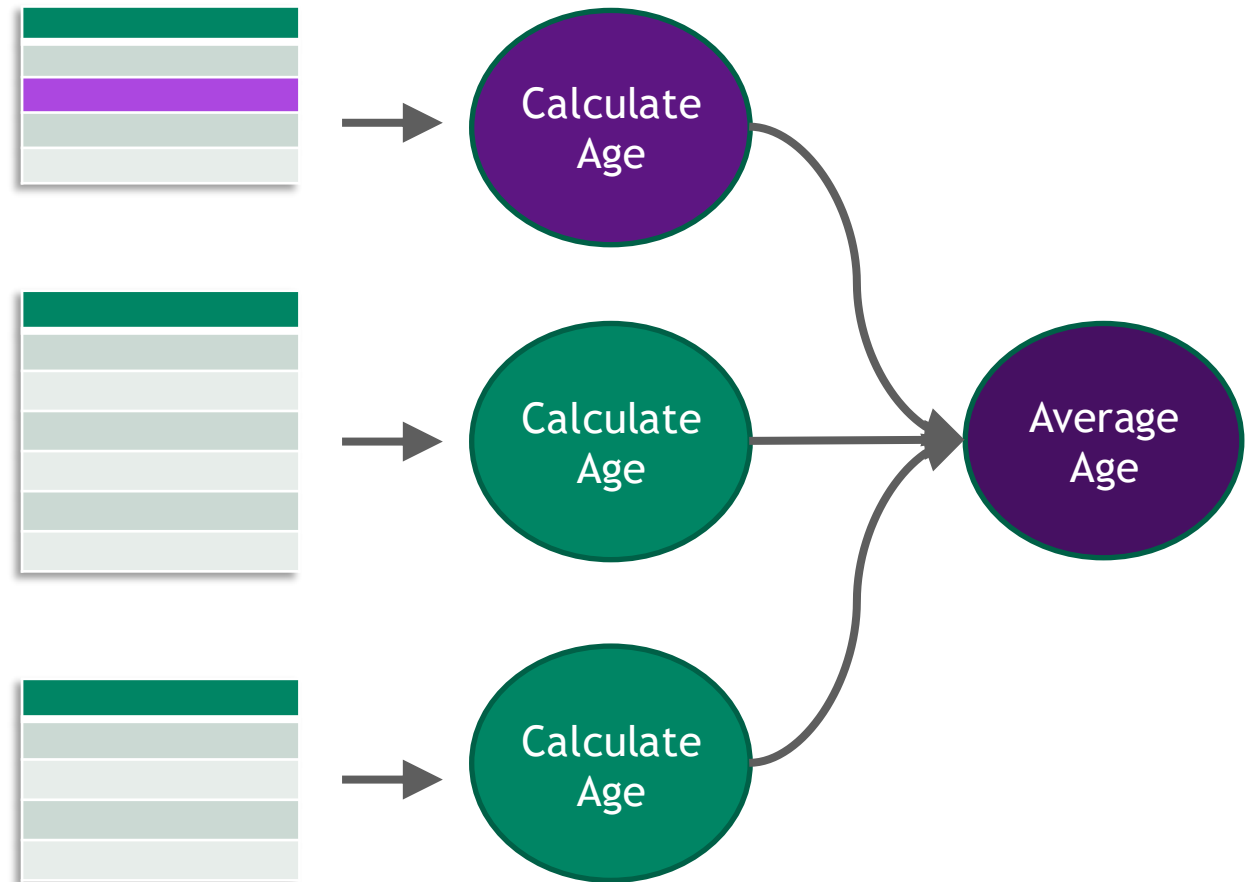
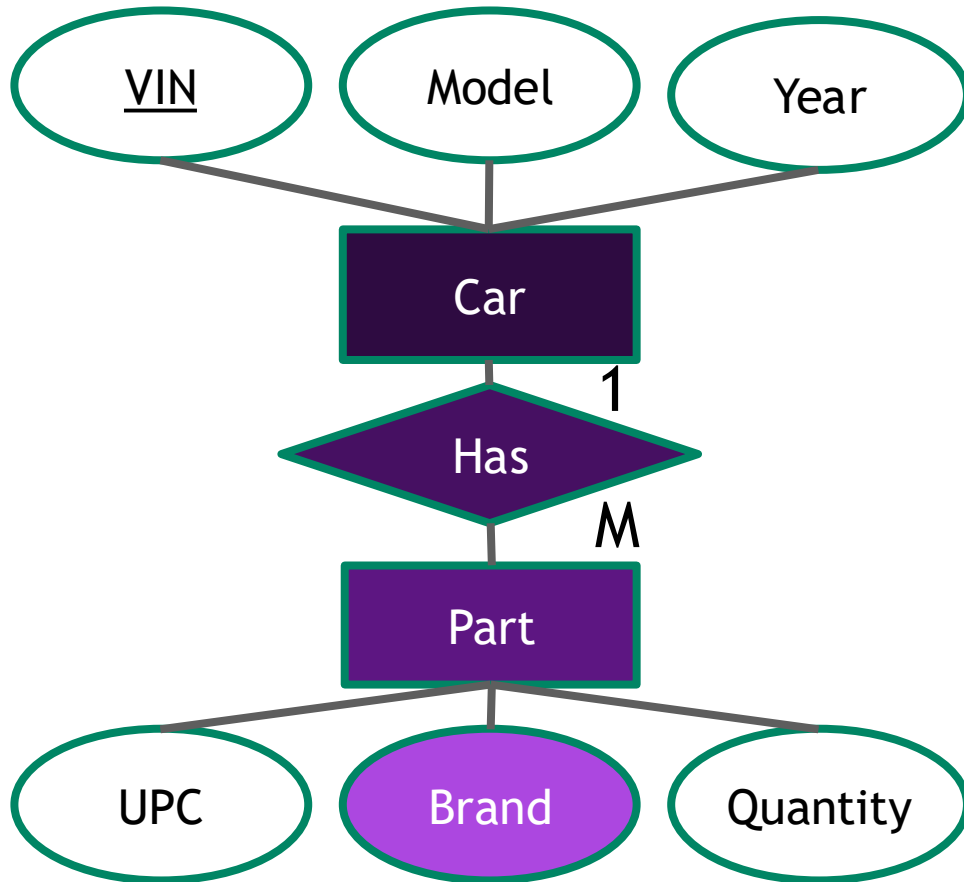
DATABASE SYSTEM DESIGN

Directed Acyclic Graphs



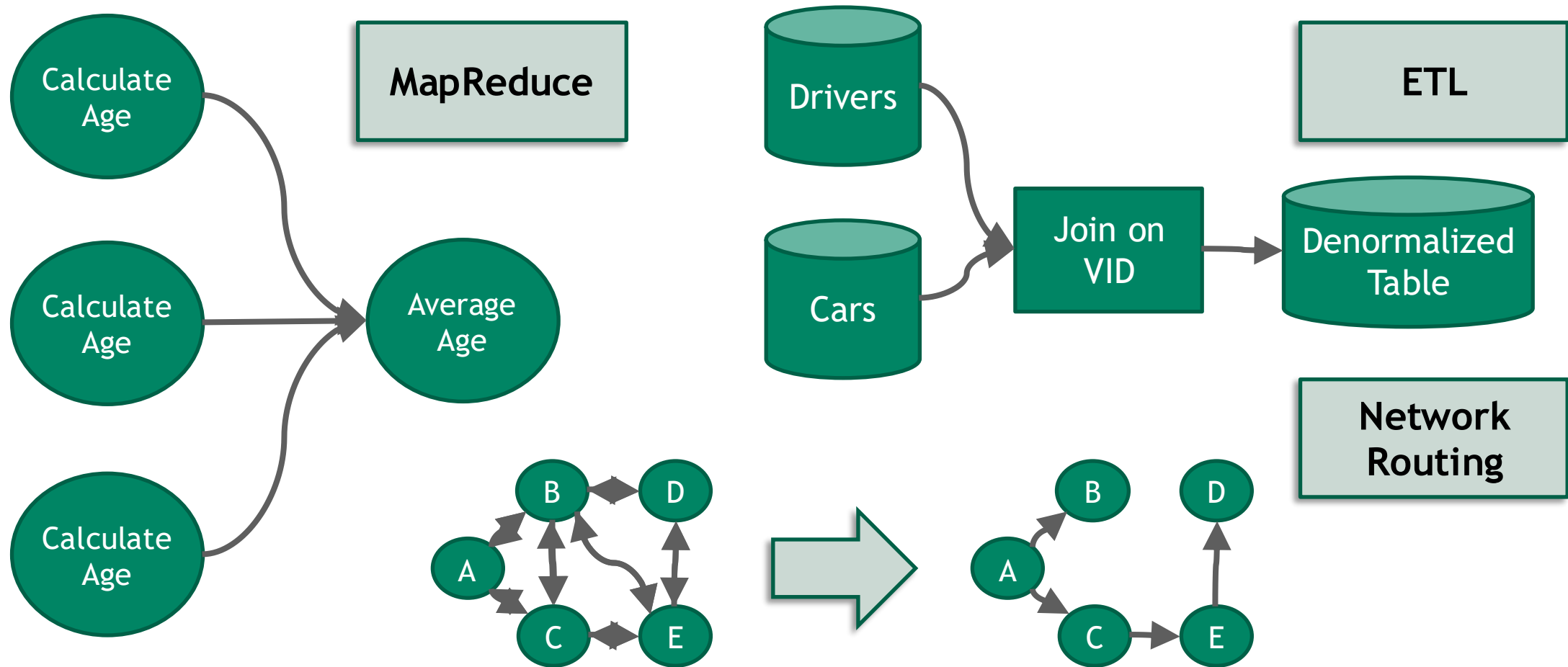
DATABASE SYSTEM DESIGN

Data Quality



DATABASE SYSTEM DESIGN

Directed Acyclic Graphs

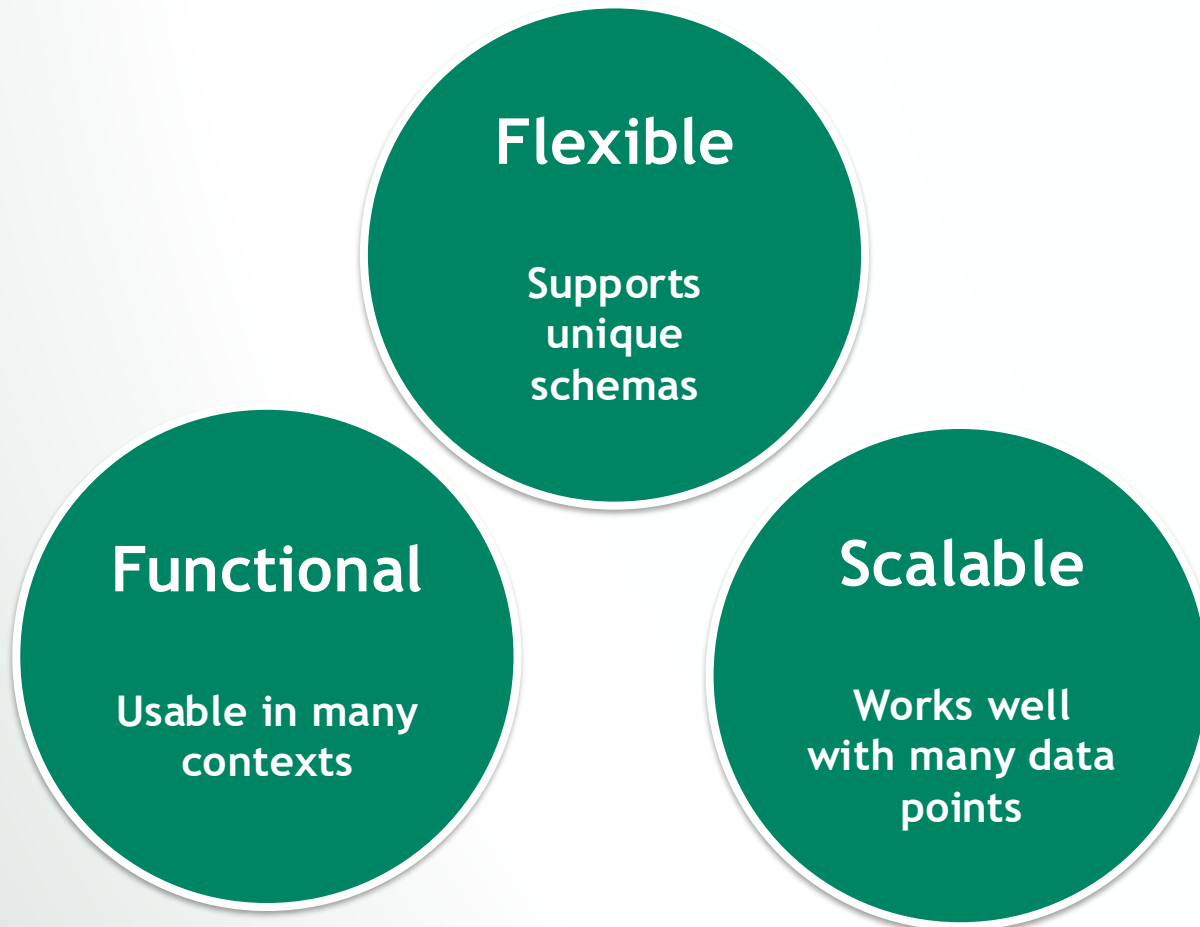




FILE FORMATS

DATA FORMATS

Pick 1 - 2



These definitions vary based on context:

- Scalable to read or to write?
- Scalable with speed or cost?
- Flexible in the data store or flexible in the application?
- Functional for the server or functional for the client?

PICKING THE BEST FORMAT

CRUD

- Create
 - Add a record
- Read
 - Get record
- Update
 - Change a record
- Delete
 - Remove a record



ROW VS COLUMNAR STORAGE

- Row -

Efficient for
Adding a new record

-
- Formats
 - CSV (Comma-Separated Values)
 - TSV (Tab-Separated Values)
 - Apache AVRO
 - Engines
 - MySQL
 - PostgreSQL

| Columnar |

Efficient for
Data Aggregation

-
- Formats
 - Apache Parquet
 - Engines
 - BigQuery
 - Snowflake
 - Redshift

WRITING

Adding a New Entry

Row Formatted Data

Grace | Hopper | 1906

Blaise | Pascal | 1623

Katherine | Johnson | 1918

Alan | Turing | 1912

Can concatenate to end, or
inserted by row number

First	Last	Born
Grace	Hopper	1906
Blaise	Pascal	1623
Katherine	Johnson	1918

+

Alan	Turing	1912
------	--------	------

Column Formatted Data

Grace | Blaise | Katherine

Hopper | Pascal | Johnson

1906 | 1623 | 1918

Alan

Turing

1912

Broken up and inserted at
the end of each block

ANALYSIS

A.K.A Feature Engineering

Row Formatted Data

Grace | Hopper |

Blaise | Pascal |

Katherine | Johnson |

GH

BP

KJ

Broken up and inserted at
the end of each block

First	Last	In.
Grace	Hopper	GH
Blaise	Pascal	BP
Katherine	Johnson	KJ
Alan	Turing	AT

>

Column Formatted Data

Grace | Blaise | Katherine |
Hopper | Pascal | Johnson |

GH | BP | KJ

Can be concatenated to end or
inserted by column number

BINARY

Ex: Multimedia File



Pro

- Compact
 - Faster to send and process
- Flexible
 - Many datatypes can easily be converted to binary
 - Great for images



Con

- Hard to visualize without decoding software
 - Difficult to debug data integrity

ASCII

Ex: CSV



Pro

- Simple structure
 - No file metadata
- File is human readable
- Average scalability
 - Easy to join and split multiple CSV files
 - Easy to append a new entry



Con

- Simple structure
 - No file metadata
- Average scalability
 - Data is not compressed as much as other file types

PARQUET

Ex: Hadoop



Pro

- Good compression if many repeated values
- Efficient to read a subset of columns
- Support for complex datatypes like arrays



Con

- Immutable
 - Query results are typically saved in a new file
- Querying for all the attributes of an entity is an expensive operation
- Files are not human readable without a tool

DATA FORMATS COMPARISON

Summary

Properties	CSV	JSON	Parquet	Avro
Columnar			✓	
Compressible	✓	✓	✓	✓
Splittable	✓	✓	✓	✓
Human readable	✓	✓		
Complex data structure		✓		
Schema evolution/validation		✓	✓	✓
Binary			✓	✓



DATA FRAMEWORKS

VERTICAL VS HORIZONTAL SCALING

↑ Vertical ↑

Scales to higher quality hardware

- SQL
- CuPy
- NumPy
- cuDF
- pandas

← Horizontal →

Scales to more partitions / machines

- Dask
- NoSQL
- Spark
- Hadoop

SQL

Structured Query Language

Query

```
SELECT first, last  
FROM awesome.people  
WHERE born > 1900
```

Table

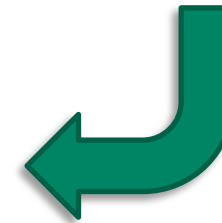
(awesome.people)

first	last	born
Grace	Hopper	1906
Blaise	Pascal	1623
Katherine	Johnson	1918
Alan	Turing	1912



first	last
Grace	Hopper
Katherine	Johnson
Alan	Turing

Result



DATAFRAMES

Pandas (CPU) and cuDF (GPU)

Query



```
df = df[df["born"] > 1900]
```

```
df = df["first", "last"]
```

Table

df

first	last	born
Grace	Hopper	1906
Blaise	Pascal	1623
Katherine	Johnson	1918
Alan	Turing	1912



first	last
Grace	Hopper
Katherine	Johnson
Alan	Turing

Result



MATRICES AND NUMBER ARRAYS

NumPy (CPU) and CuPy (GPU)

Query



```
a = a.sum(axis = 0)
```

Array

a

8	6	7
5	3	0
9	8	6
7	5	3

Result

29

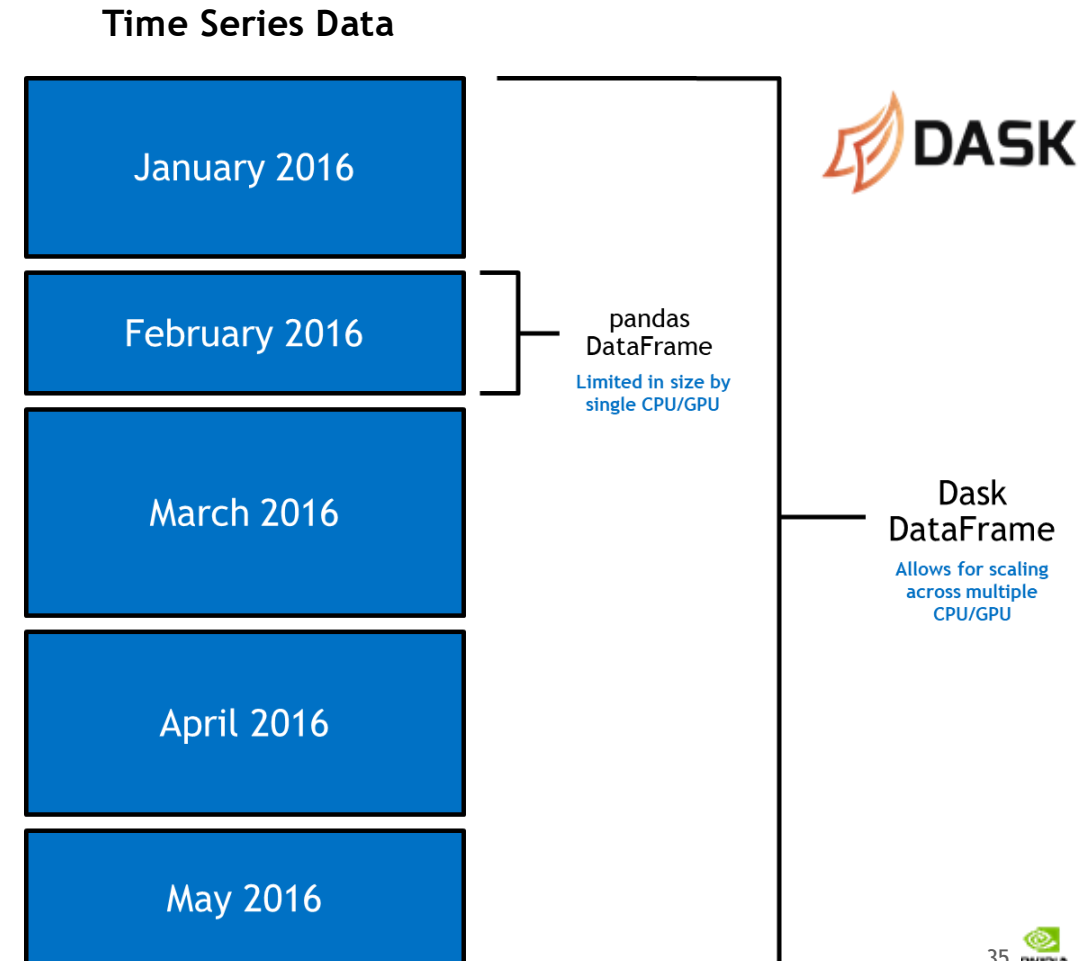
22

16

DASK SCALES PYTHON ANALYTICS

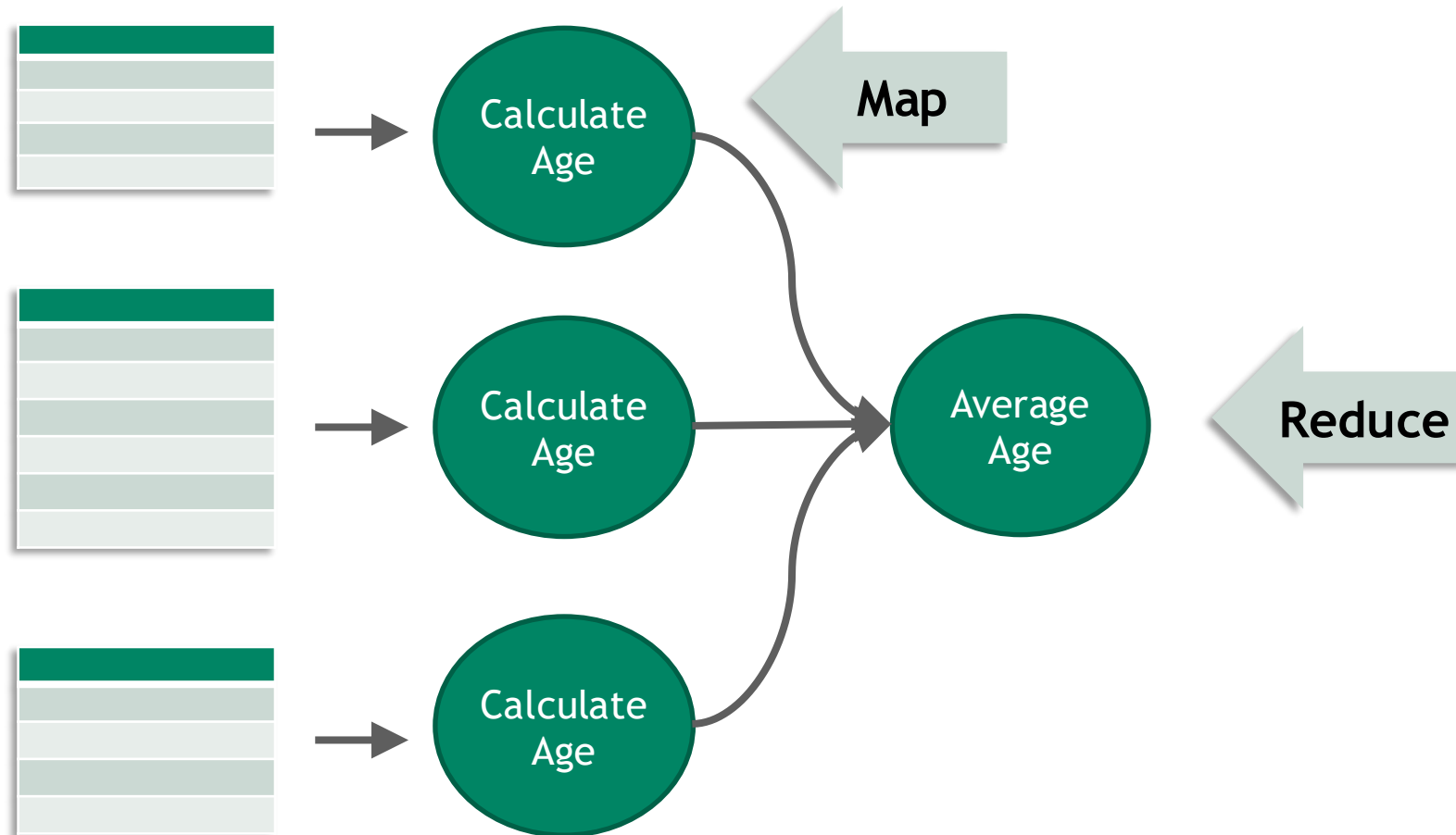
SCALE FROM A LAPTOP TO LARGE-SCALE CLUSTERS WITH EASE

- **Dask** enables data scientists to scale out analytics workloads in native Python. With an optimized scheduler, Dask makes it easy to schedule and execute tasks on distributed computation.
- **Dask** follows the standards set by the PyData ecosystem to provide a familiar, comfortable user experience at scale. When paired with NVTABULAR/RAPIDS, data scientists can leverage the processing power of NVIDIA accelerated compute and distribute across clusters to improve cycle time-reducing time to insights drastically.



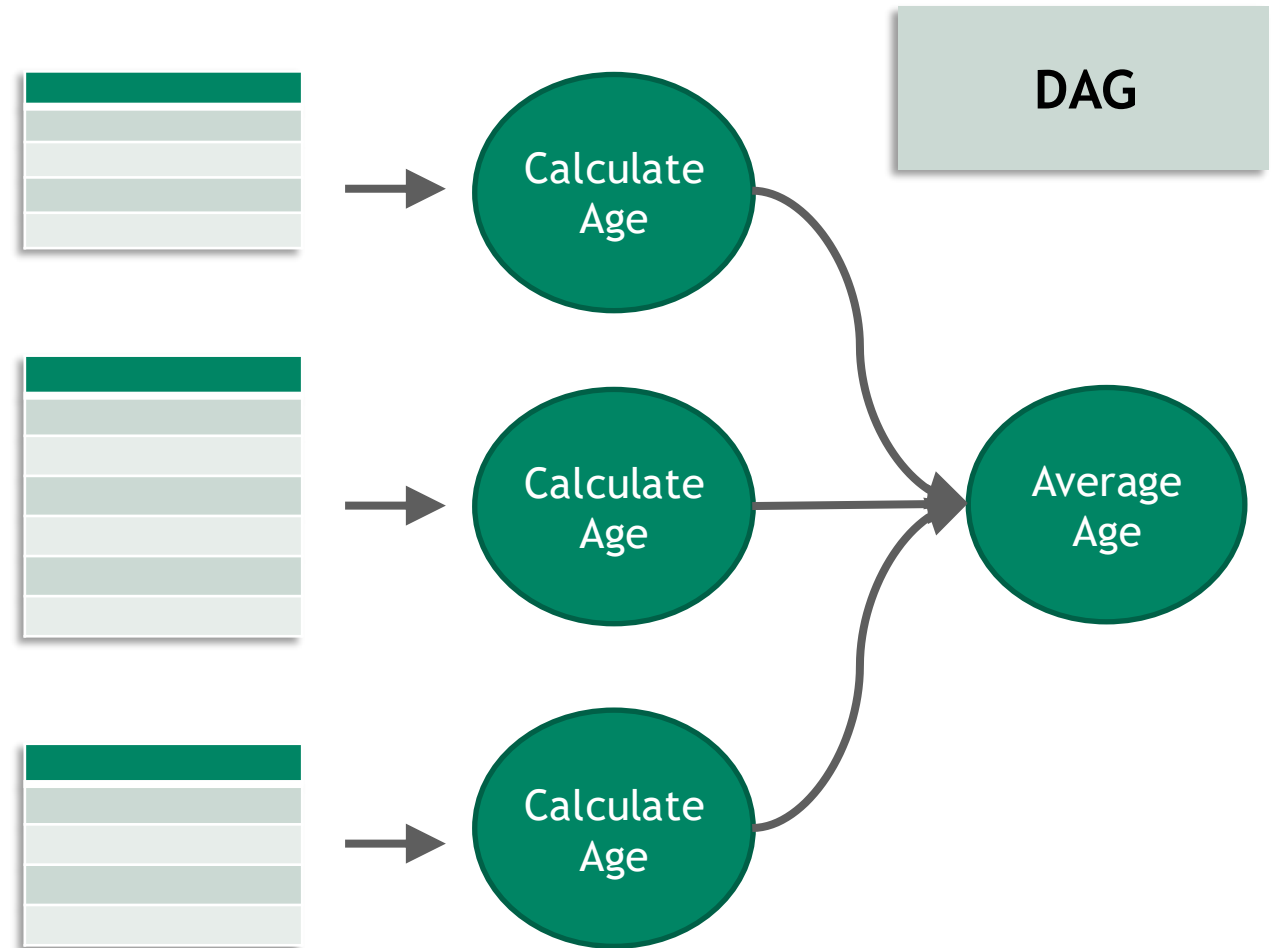
MAPREDUCE

Map to each thread, Reduce all threads to one



LAZY EXECUTION

Building a Factory



RELATIONAL DATABASES

Ex: SQL



Pro

- Well known
- Concise Language
- Relatively fast querying
 - Foreign keys
- Blazing SQL



Con

- Inflexible data structure
 - Some objects do not convert well to table format
- Typically, single server
 - More expensive hardware needed to scale

DATAFRAME

Ex: cuDF, Pandas, R



Pro

Python and R APIs

- cuDF, Pandas
- Compared to SQL, more flexible operations
- Easier to make user-defined functions and integrate third party libraries



Con

- Single server, not meant for large-scale data manipulation
 - Consider Spark instead
- Compared to SQL, not as scalable

DASK

Ex: Dask DataFrame, Dask-cuDF



Pro

- Large computation can receive a significant speed increase
- Can read large data sources due to partitioning



Con

- Large overhead to set up not worth it for small files or limited computation
- Lazy execution can make it tricky to debug



LAB

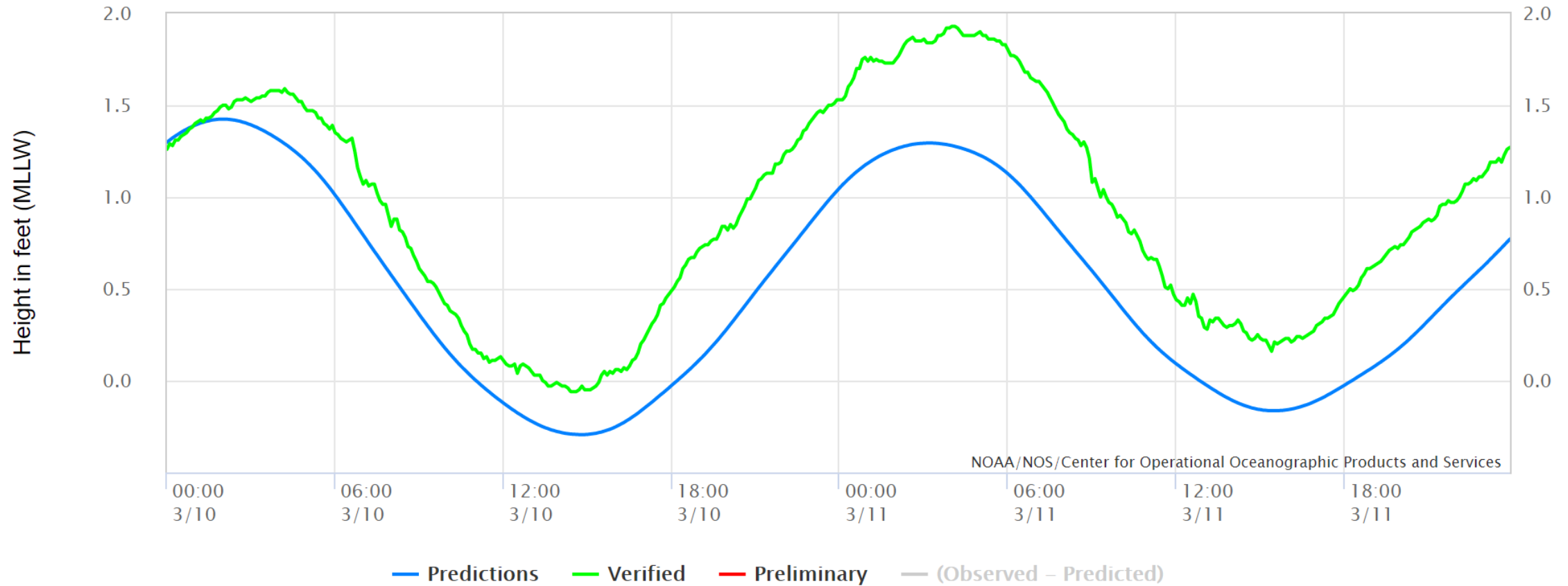
WEATHER SYSTEMS



Credit: Ralph F. Kresge, Submitted to NOAA

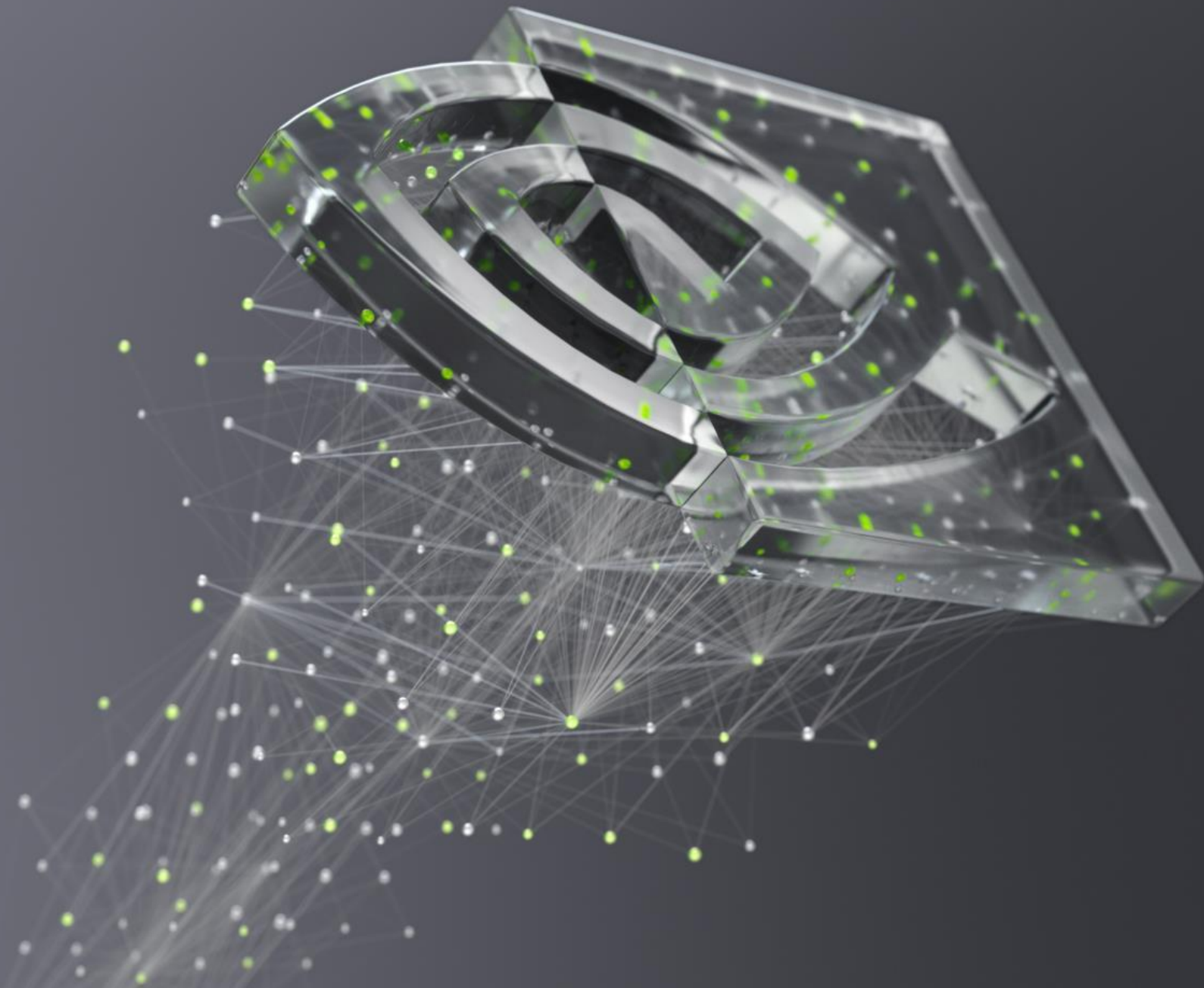
INVESTIGATING WATER LEVEL

NOAA/NOS/CO-OPS
Observed Water Levels at 8735523, East Fowl River Bridge AL
From 2021/03/10 00:00 GMT to 2021/03/11 23:59 GMT





LET'S GO!



DEEP
LEARNING
INSTITUTE