Big-data Systems: A Tour

Ahmed Eldawy

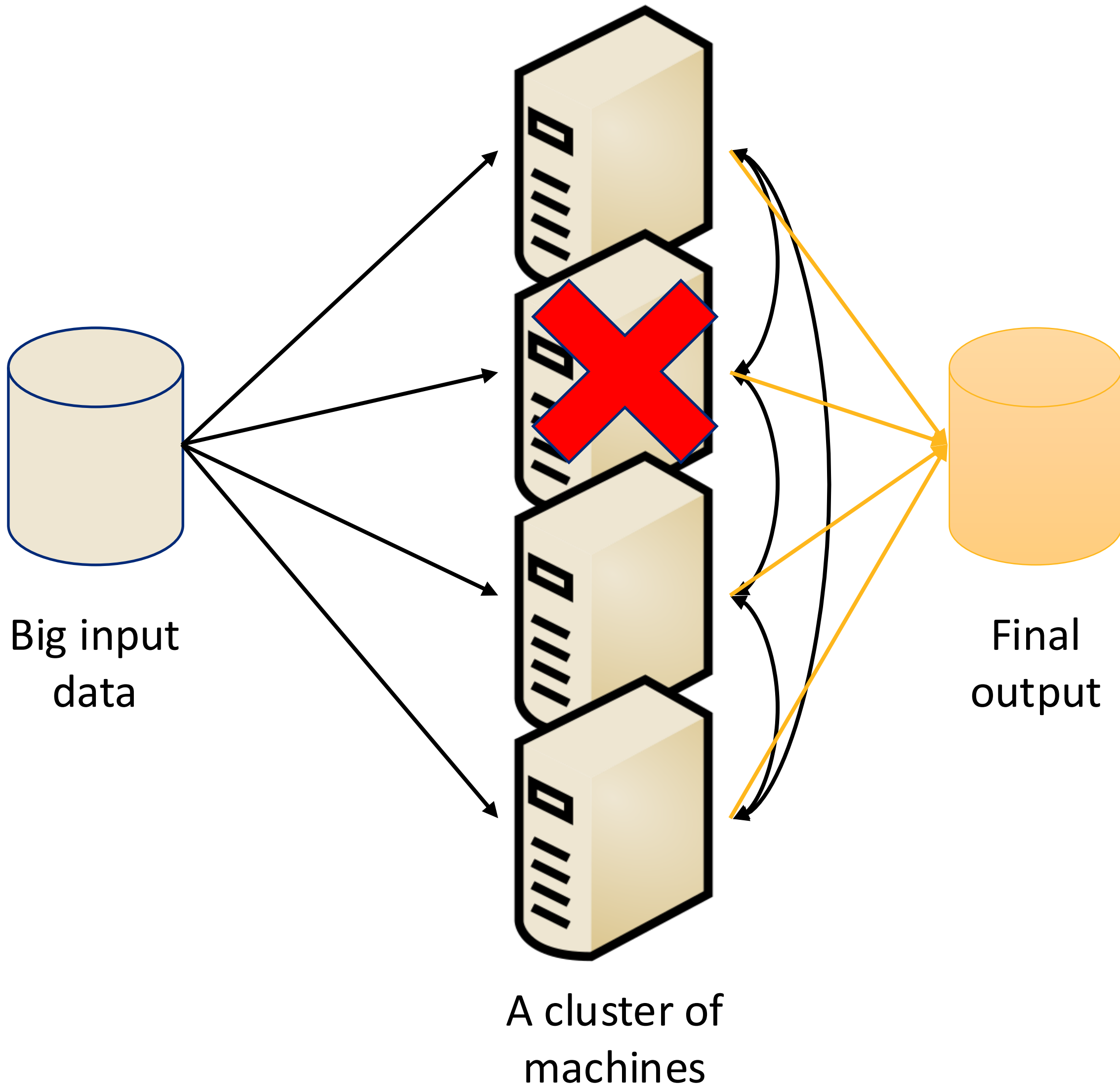
CELEBRATING 30 YEARS
Marlan and Rosemary Bourns
College of Engineering

Distributed Data Processing

- The idea of distributed databases is older than you might think
  Richard Peebles, Eric G. Manning: A Computer Architecture for Large (Distributed) Data Bases. **VLDB 1975**: 405-427
- Distributed data structures and algorithms have always been around
- So, what is new?

CELEBRATING 30 YEARS
Marlan and Rosemary Bourns College of Engineering

Distributed Data Processing



Data partitioning
Load balancing
Fault tolerance
Synchronization

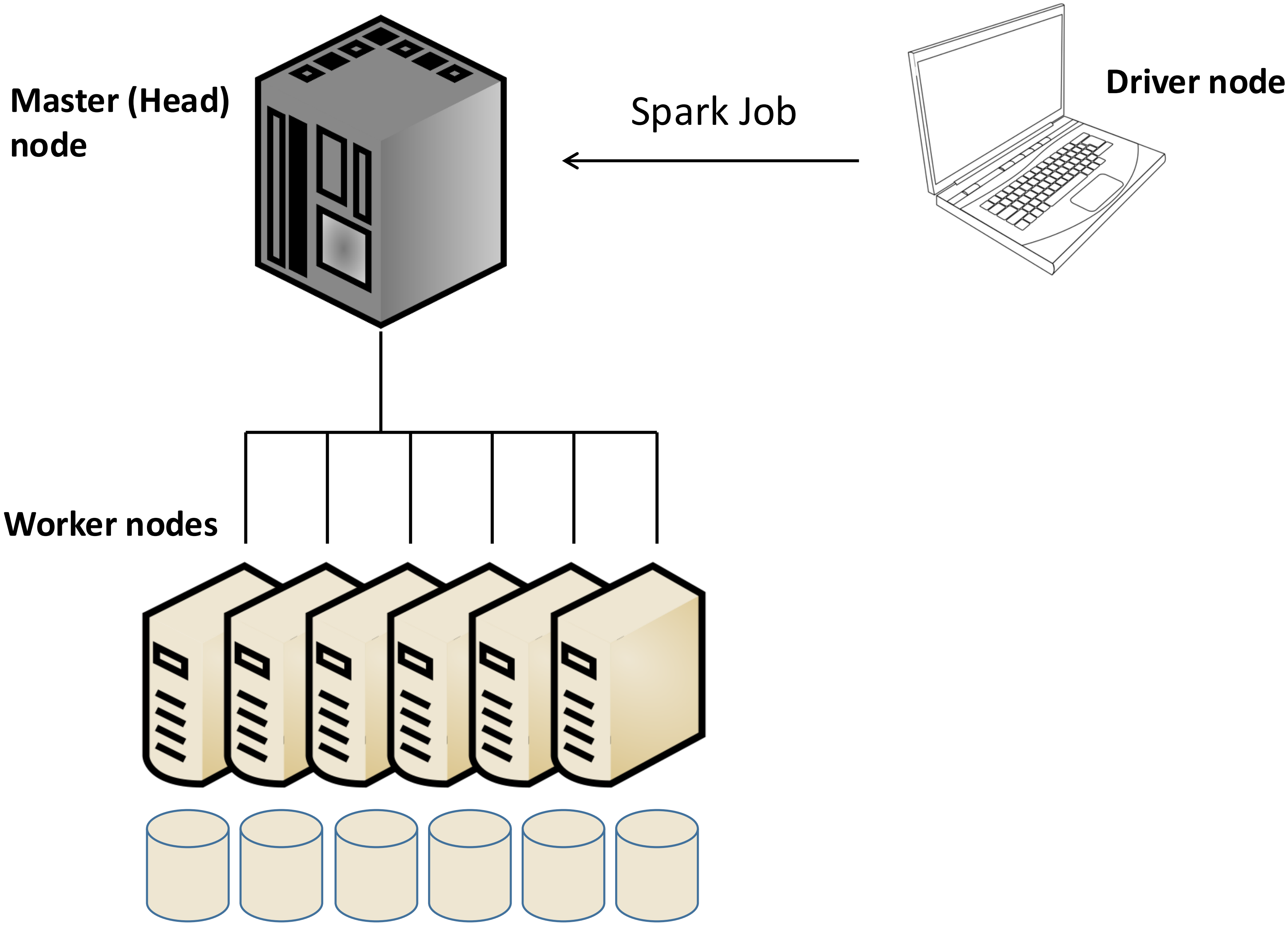
MapReduce

- A programing paradigm for expressing distributed algorithms
- Introduced by Google in 2004
  - Google File System for distributed storage
  - Google MapReduce for distributed processing
- Hadoop is the open-source counterpart released in 2007 and contributed mainly by Yahoo!
  - HDFS
  - Hadoop MapReduce

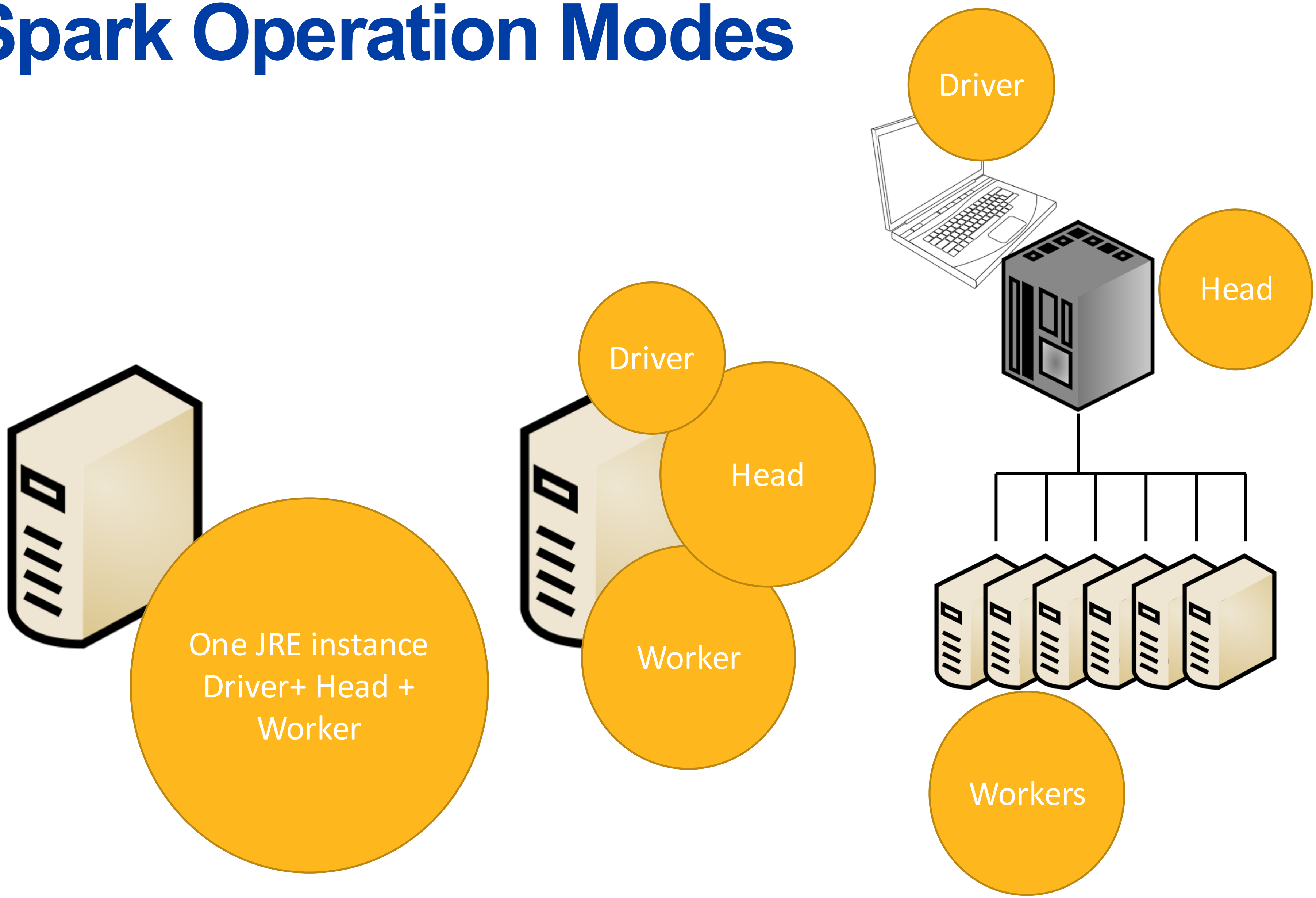
Spark

- Hadoop and MapReduce were perfect as research vehicles
- They helped in framing what we really want in a big data system
- Spark came as a new system designed from scratch to satisfy the real need of big data
- A distributed shared-nothing system
- Uses a functional programming paradigm

Spark Overview



Spark Operation Modes



Local mode Stand-alone mode Cluster mode

Examples

Examples

host	time	method	url	response	bytes
pppa006.compuserve.com	807256800	GET	/images/launch-logo.gif	200	1713
vcc7.langara.bc.ca	807256804	GET	/shuttle/missions/missions.html	200	8677

```
# Initialize the Spark context
JavaSparkContext spark =
    new JavaSparkContext("local", "CS226-Demo");
```

Examples

```
// Initialize the Spark context
JavaSparkContext spark =
    new JavaSparkContext("local", "CS226-Demo");

// Hello World! Example. Count the number of lines in the file
JavaRDD<String> textFileRDD =
                  spark.textFile("nasa.tsv");
long count = textFileRDD.count();
System.out.println("Number of lines is "+count);
```

Examples

```
// Count the number of OK lines
JavaRDD<String> okLines =
 textFileRDD.filter(s -> s.split("\t")[5].equals("200"));

long count = okLines.count();
System.out.println("Number of OK lines is "+count);
```

(Semi-) Structured Data Processing using SparkSQL

CELEBRATING 30 YEARS
UCR Marlan and Rosemary Bourns College of Engineering

Structured Data Processing

- A common use case in big-data is to process structured or semi-structured data
- In Spark RDD, all functions and objects are black-boxes.
- Any structure of the data has to be part of the functions which includes:
  - Parsing
  - Conversion
  - Processing

SparkSQL

- Redesigned to consider Spark query model
- Supports all the popular relational operators
- Can be intermixed with RDD operations
- Uses the Dataframe API as an enhancement to the RDD API

Dataframe = RDD + schema

Built-in operations in SprkSQL

- Filter (Selection)
- Select (Projection)
- Join
- GroupBy (Aggregation)
- Load/Store in various formats
- Cache
- Conversion between RDD (back and forth)

SparkSQL Examples

Code Setup

```
SparkSession sparkS = SparkSession
        .builder()
        .appName("Spark SQL examples")
        .master("local")
        .getOrCreate();

Dataset<Row> log_file = sparkS.read()
        .option("delimiter", "\t")
        .option("header", "true")
        .option("inferSchema", "true")
        .csv("nasa_log.tsv");
log_file.show();
```

Filter Example

```
// Select OK lines
Dataset<Row> ok_lines =
log_file.filter("response=200");
long ok_count = ok_lines.count();
System.out.println("Number of OK lines is
"+ok_count);


// Grouped aggregation using SQL
Dataset<Row> bytesPerCode =
log_file.sqlContext().sql("SELECT response,
sum(bytes) from log_lines GROUP BY response");
```

MLlib: Machine learning in Spark

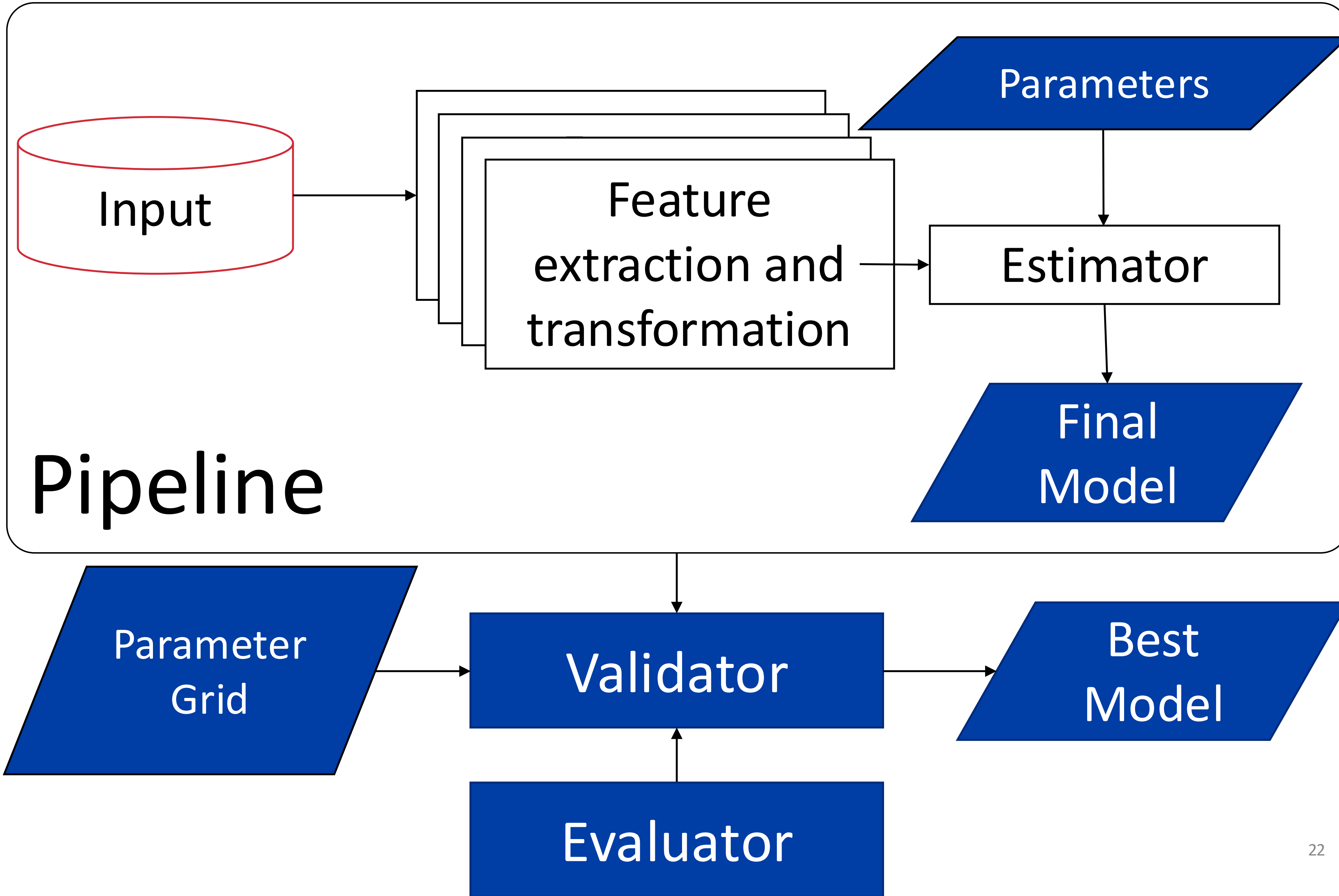
Machine Learning Algorithms

- Supervised learning
  - Given a set of features and labels
  - Builds a model that predicts the label from the features
  - E.g., classification and regression
- Unsupervised learning
  - Given a set of features without labels
  - Finds interesting patterns or underlying structure
  - E.g., clustering and association mining

Basic Statistics

- Column statistics
  - Minimum, Maximum, count, … etc.
- Correlation
  - Pearson’s and Spearman’s correlation
- Hypothesis testing
  - Chi-square Test χ^2

ML Pipeline


Parameters
Input
Feature extraction and transformation
Estimator
Final Model
Pipeline
Parameter Grid
Validator
Best Model
Evaluator

Code Example on MLlib

Input Data

House ID	Bedrooms	Area (sqft)	…	Price
1	2	1,200		$200,000
2	3	3,200		$350,000
…				

- Goal: Build a model that estimates the price given the house features, e.g., # of bedrooms and area

Initialization

- Similar to SparkSQL

```
val spark = SparkSession
  .builder()
  .appName(”SparkSQL Demo")
  .config(conf)
  .getOrCreate()

// Read the input
val input = spark.read
  .option("header", true)
  .option("inferSchema", true)
  .csv(inputfile)
```

Transformations

```
// Create a feature vector
val vectorAssembler = new VectorAssembler()
  .setInputCols(Array("bedrooms", "area"))
  .setOutputCol("features")

val linearRegression = new LinearRegression()
  .setFeaturesCol("features")
  .setLabelCol("price")
  .setMaxIter(1000)
```

Create a Pipeline

```
val pipeline = new Pipeline()
  .setStages(Array(vectorAssembler, linearRegression))

// Hyper parameter tuning
val paramGrid = new ParamGridBuilder()
  .addGrid(linearRegression.regParam,
      Array(0.3, 0.1, 0.01))
  .addGrid(linearRegression.elasticNetParam,
      Array(0.0, 0.3, 0.8, 1.0))
  .build()
```

Cross Validation

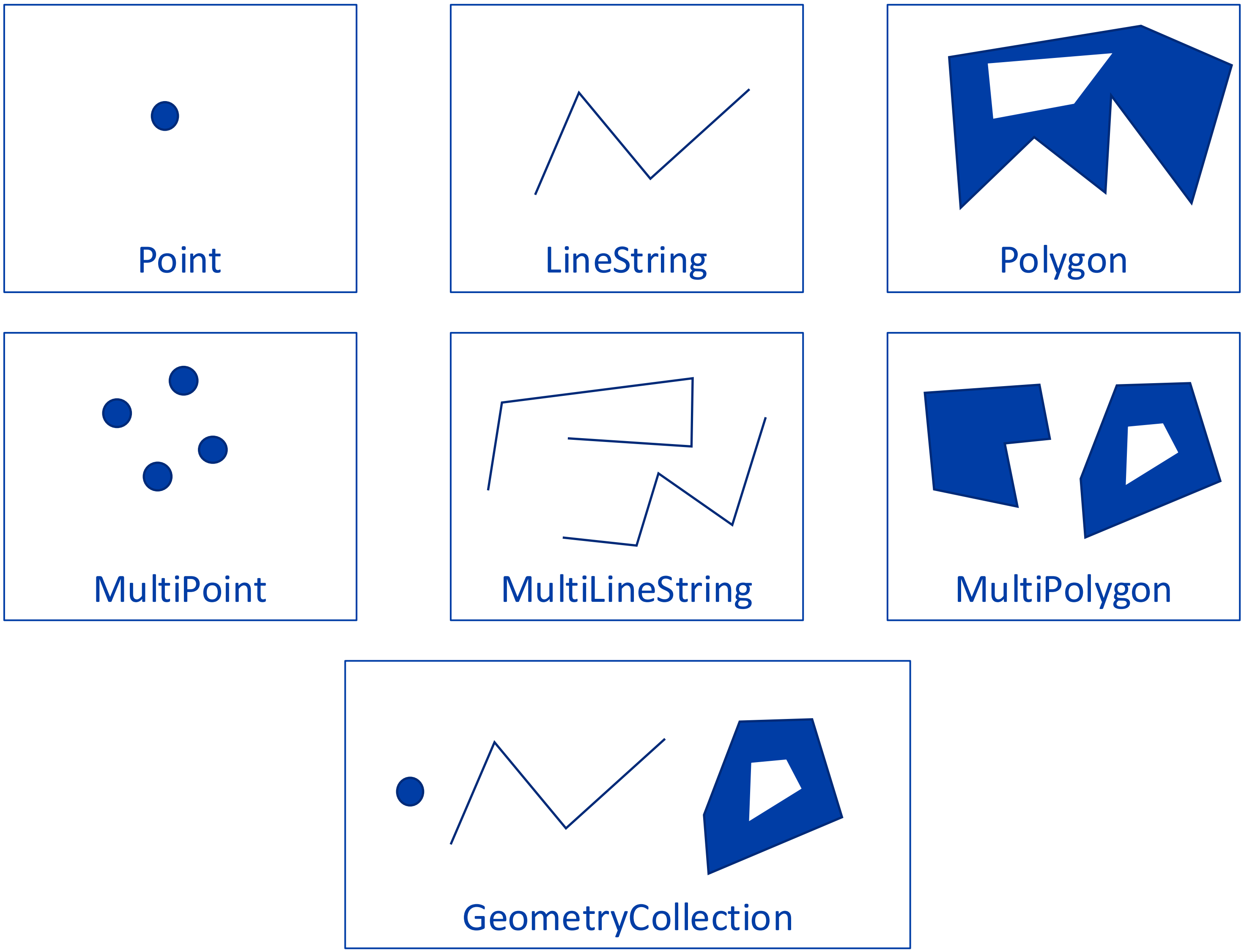
```
val crossValidator = new CrossValidator()
  .setEstimator(pipeline)
  .setEvaluator(new
RegressionEvaluator().setLabelCol("price"))
  .setEstimatorParamMaps(paramGrid)
  .setNumFolds(5)
  .setParallelism(2)

val Array(trainingData, testData) =
input.randomSplit(Array(0.8, 0.2))
val model = crossValidator.fit(trainingData)
```

Apply the model on test data

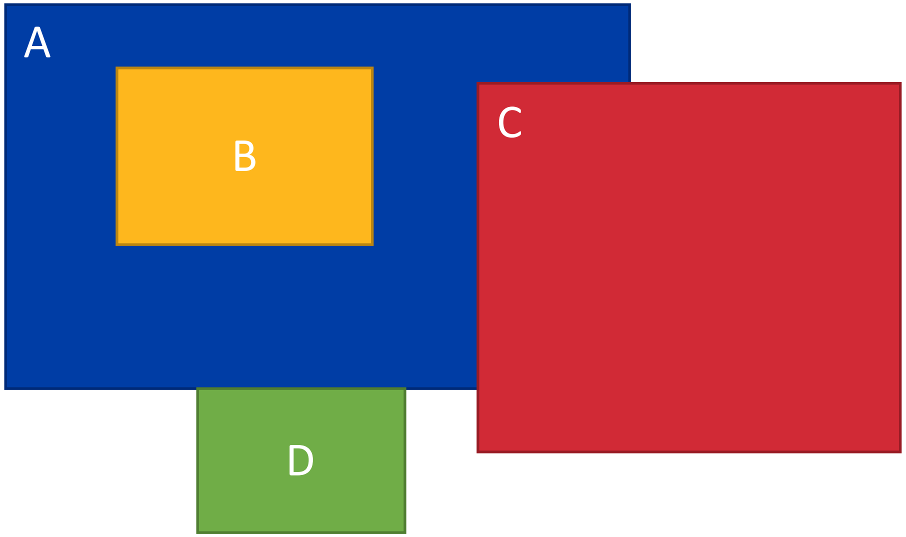
```scala
val predictions = model.transform(testData)
// Print the first few predictions
predictions.select("price", "prediction").show(5)

val rmse = new RegressionEvaluator()
  .setLabelCol("price")
  .setPredictionCol("prediction")
  .setMetricName("rmse")
  .evaluate(predictions)
println(s"RMSE on test set is $rmse")
```

Big Spatial Data Management on Spark

Geometry Data Types



Geometry Predicates



A Contains B

A Overlaps C

B Disjoint C

A Touches D

Spatial Feature (IFeature)

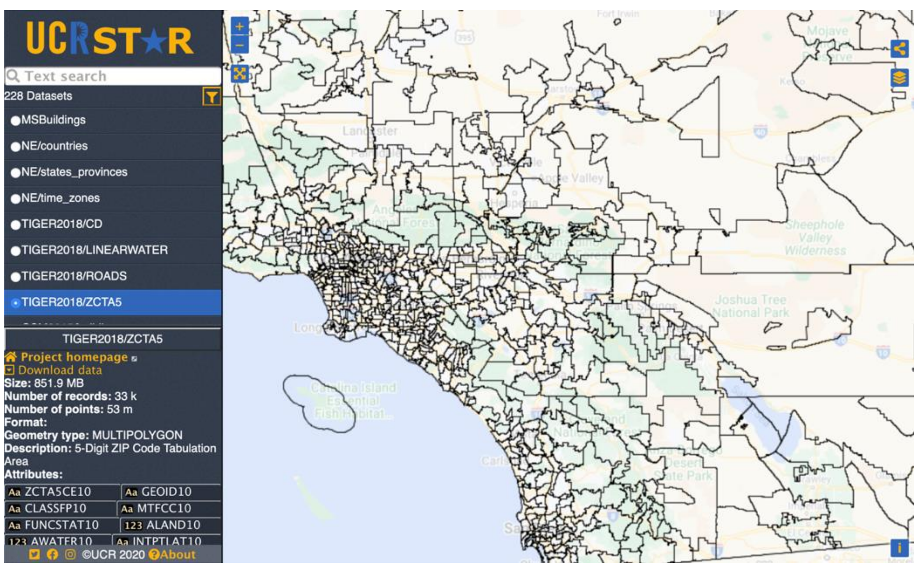
Feature = Geometry + Other Attributes

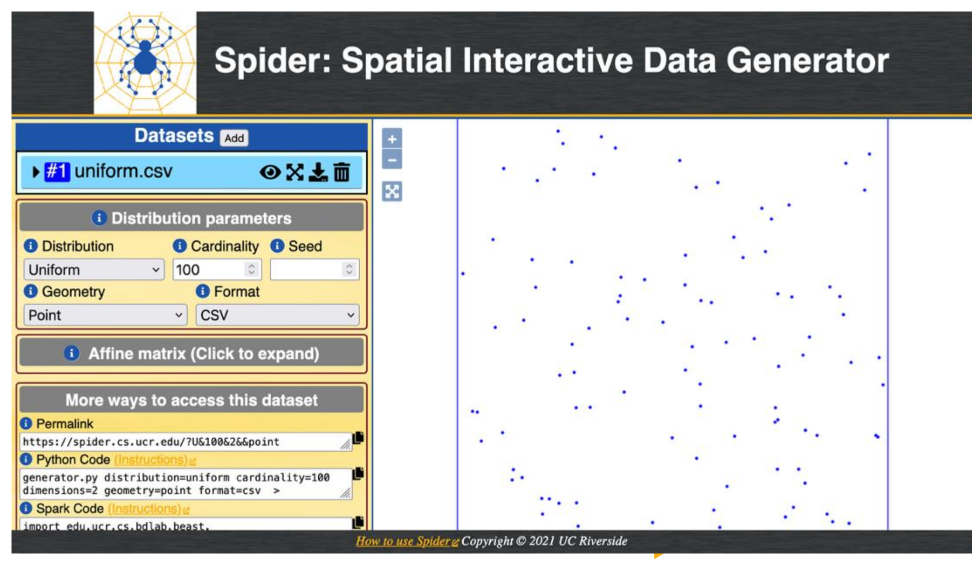
- Example
  - Road(Geometry, Name, Speed Limit)
  - State(Geometry, Name, Population)

Data Source

- UCRStar.com
- 200+ datasets
- Full/subset download
- Standard formats

- spider.cs.ucr.edu
- Data generator





Data Loading

```
// Load a shapefile
val polygons: RDD[IFeature] =
sc.shapefile("tl_2018_us_state.zip")
// Load GeoJSON file
val points = sc.geojsonFile("Tweets.geojson")

// Load points from a CSV file
val lines = sc.readCSVPoint("Crimes.csv",
     "Longitude", "Latitude", ',', skipHeader
= true)

// Load geometries from a CSV file
val lines = sc.readWKTFile("States.csv", 0,
     '\t', skipHeader = false)
```

Simple Manipulation

```
// Calculate the area and append as a new
attribute
polygons.map(f => Feature.append(f,
  value = f.getGeometry.getArea, name = "area"))

// Simplify the geometries into their convex hull
polygons.map(f => {
  val convex_hull = f.getGeometry.convexHull()
  Feature.create(f, geometry = convex_hull)
})
```

Range Filters

```
// Select the geometry of the state of California
val california: IFeature = polygons.filter(f =>
f.getAttributeValue("NAME") == "California").first()

// Filter the points that are inside the state of
California
val californiaPoints =
points.rangeQuery(california.getGeometry)
println(s"Number of points in California
${californiaPoints.count()}")
```

```
Output


Number of points in California 259657
```

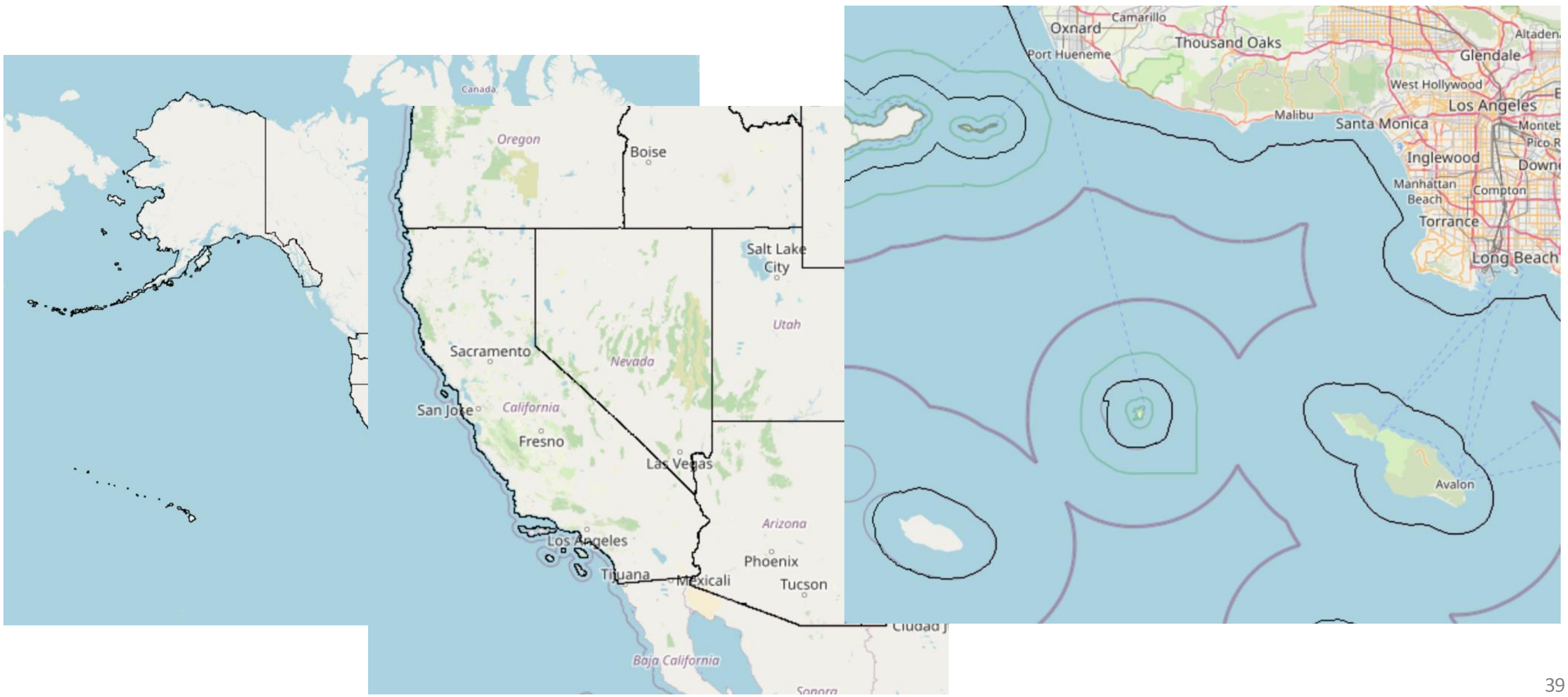
Spatial Join

```
// Count points per state
val airportCountByState =
polygons.spatialJoin(airports)
  .map(fv => (fv._1.getAs[String]("NAME"), 1))
  .countByKey()

airportCountByState.foreach(sv =>
println(s"${sv._1}\t${sv._2}"))
```

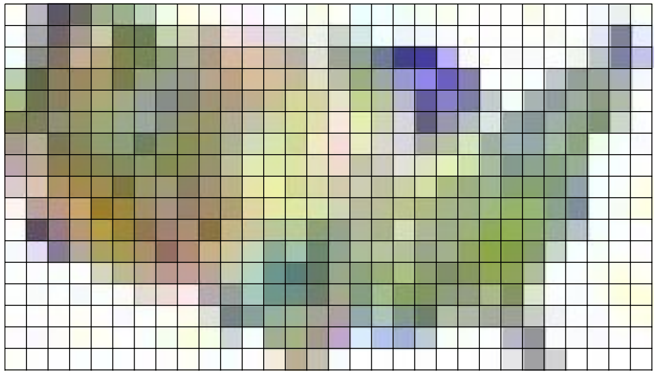
```
Output
New Mexico	1
Connecticut1
Commonwealth of the Northern Mariana Islands	2
California	12
Nevada 3
```

Visualization on a Map

```
// Plot states as a multilevel map
polygons.plotPyramid("states", 10,
   opts = "mercator" -> "true")
```

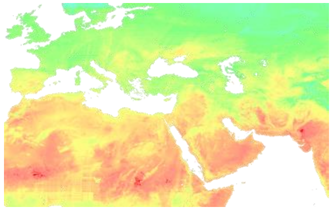


Raster Data Representation



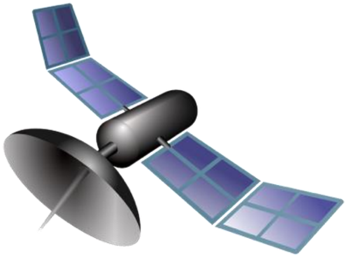
2D Array of values (pixels)

Vegetation



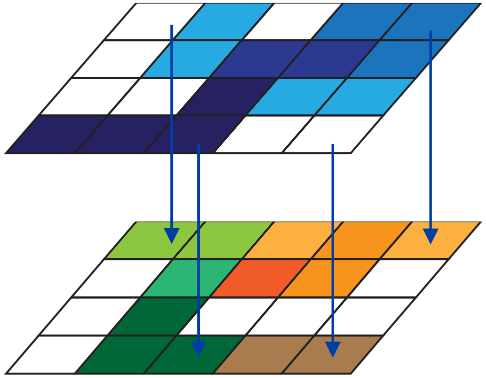
Temperature

Camera



Satellite

Map Algebra



Linear algebra operations
Map Algebra
Overlay
Rescale

Raster (Satellite) Data Processing

```
// Load a raster file (or directory)
val raster: RDD[ITile[Int]] =
sc.geoTiff[Int]("glc2000_v1_1.tif")

// Load temperature in Kelvin from an HDF file
val temperatureK: RasterRDD[Float] =
sc.hdfFile("MOD11A1.A2022173.h08v05.006.2022174092443.hdf"
, "LST_Day_1km")

// Convert Kelvin to Fahrenheit
val temperatureF: RasterRDD[Float] =
temperatureK.mapPixels(k => (k-273.15f) * 9 / 5 + 32)

// Save the result as a GeoTIFF file
temperatureF.saveAsGeoTiff("temperature_f")
```

Raster: Filter Pixels

```
val temperatureK: RasterRDD[Float] =

sc.hdfFile("MOD11A1.A2022173.h08v05.006.2022174092443
.hdf", "LST_Day_1km")

// Keep only pixels with temperature > 300°K
temperatureK.filterPixels(_>300)
    .saveAsGeoTiff("temperature_high")
```

Raster: Rescale and Reproject

```
val raster: RasterRDD[Int] =
sc.geoTiff[Int]("glc2000_v1_1.tif")
// Downscale a big raster to 360x180
val rescaled = raster.rescale(360, 180)
// Save as a single file
rescaled.saveAsGeoTiff("glc_small",
GeoTiffWriter.WriteMode -> "compatibility")
```

Raster: Raster-Vector Join

```
// Load a raster file of Global Land Cover
val raster: RasterRDD[Int] =
sc.geoTiff[Int]("glc2000_v1_1.tif")

// Filter the pixels that represent trees
val trees = raster.filterPixels(lc => lc >=1 && lc <= 10)

// Load all countries
val countries =
sc.shapefile("ne_10m_admin_0_countries.zip")

// Count number of tree pixels per country
val result = trees.raptorJoin(countries)
  .map(x => x.feature.getAs[String]("NAME"))
  .countByValue().toMap
```

Big Data Management & Analysis

with

Apache AsterixDB

CELEBRATING 30 YEARS
Marlan and Rosemary Bourns
College of Engineering

AsterixDB Overview

AsterixDB: “One Size Fits a Bunch!”

Wish-list:

- Able to **manage** data
- **Flexible** data model
- Full **query** capability
- Continuous data **ingestion**
- Efficient and robust **parallel** runtime
- Cost **proportional** to task at hand
- Support today’s “**Big Data** data types”

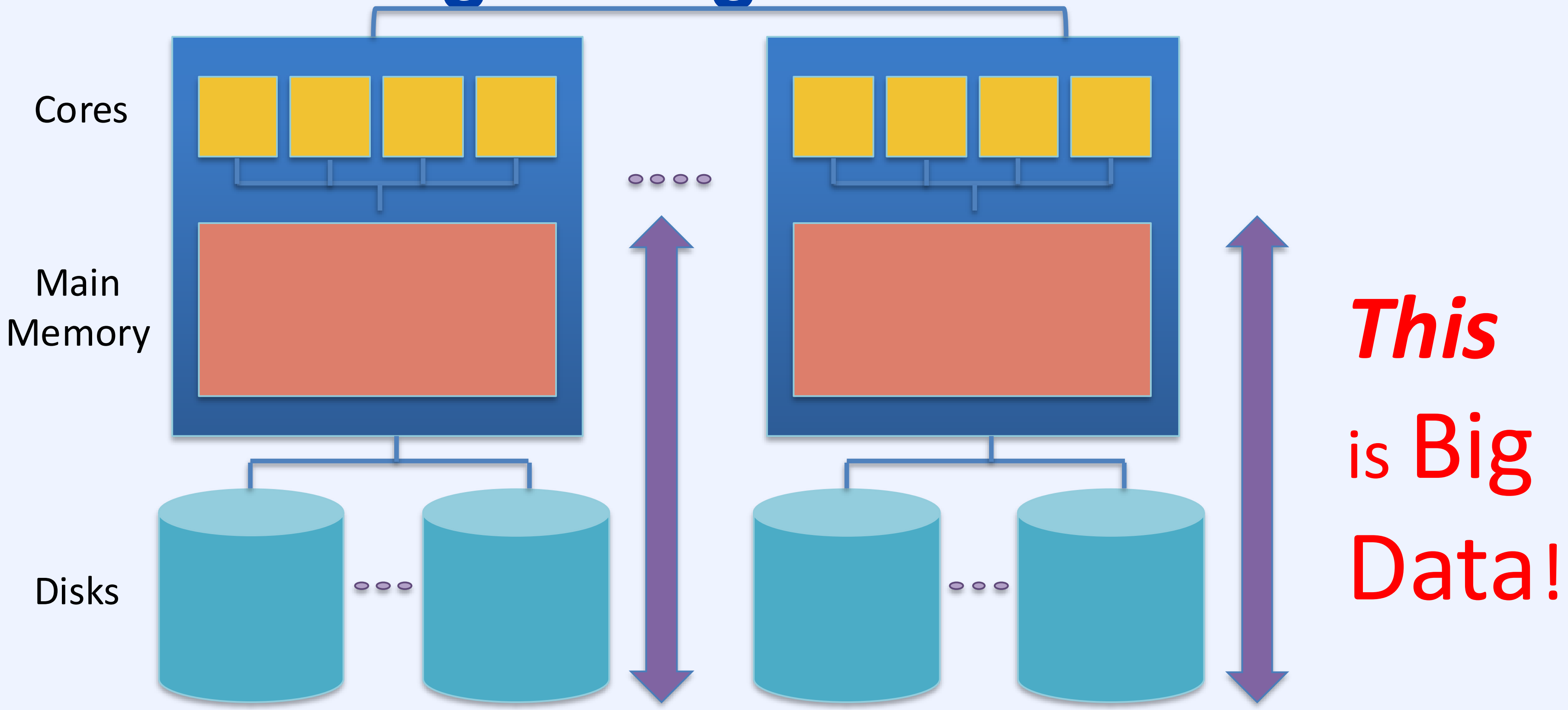
Semistructured
data management

Parallel
DB systems

First-gen BD
analysis tools

→ ***Parallel NoSQL DBMS*** ←

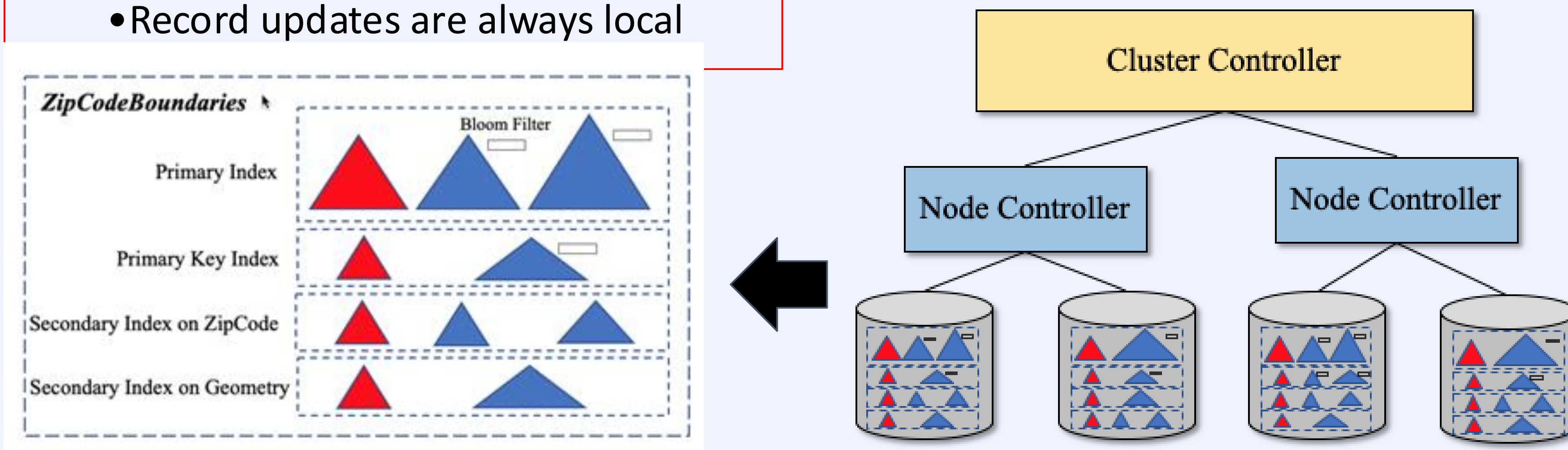
Just How Big is "Big Data"?



An Indexed Analytics Dataset

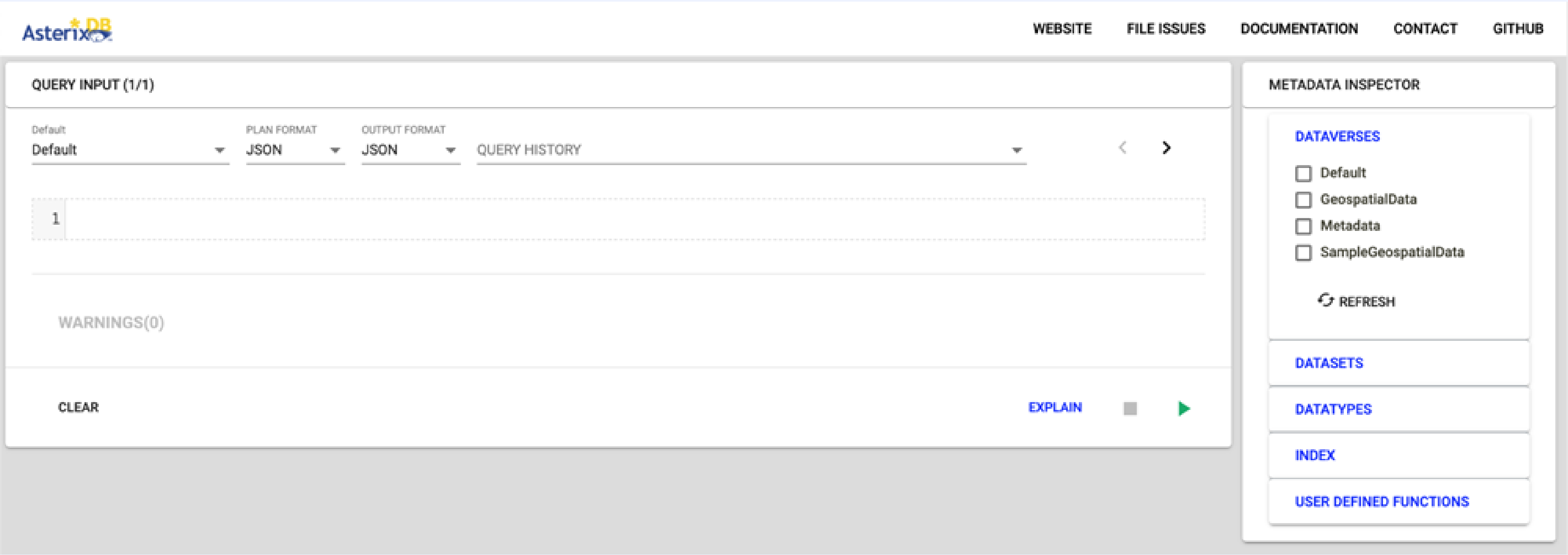
Partitioned local storage approach

- Hashed on primary key (PK)
- Primary index w/ PK + record
- Secondary index(es) with SK + PK
- Record updates are always local



Spatial Data Types and Functions

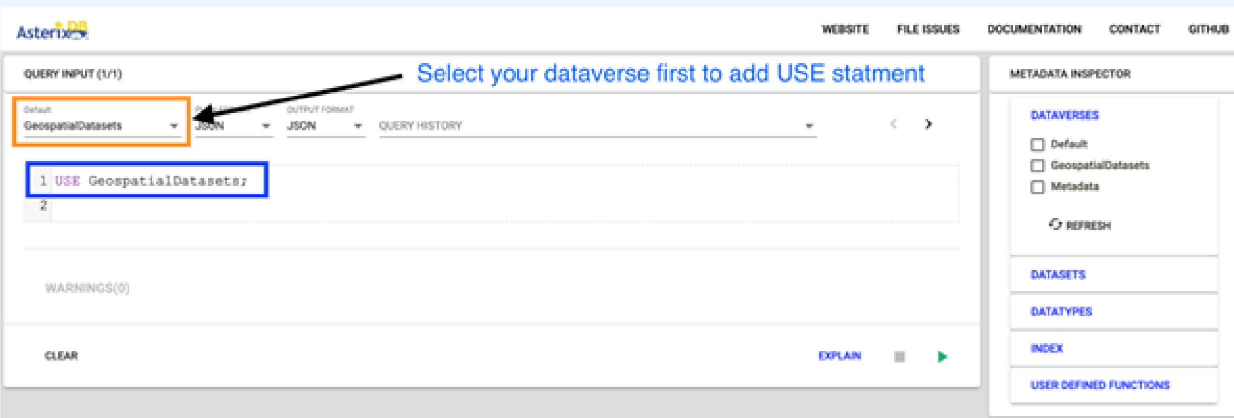
AsterixDB User Interface (http://localhost:19006)



Creating a Dataverse - Query 1

```
DROP DATAVERSE UNIQUE_DATAVERSE_NAME IF EXISTS;
CREATE DATAVERSE UNIQUE_DATAVERSE_NAME IF NOT EXISTS;
```

Please do not forget to replace UNIQUE_DATAVERSE_NAME with your



You need to select your dataset before running any query.

Loading Data to AsterixDB

```
// Chicago Crimes Dataset
CREATE TYPE ChicagoCrimesType IF NOT EXISTS AS {
  id: uuid,
  g: geometry,
  `Primary Type`: String
};

CREATE DATASET ChicagoCrimes(ChicagoCrimesType) IF
NOT EXISTS PRIMARY KEY id AUTOGENERATED;

LOAD DATASET ChicagoCrimes USING localfs
(("path"="1:///home/admin/bossdata/chicagocrimes.js
on"), ("format"="adm"));

SELECT VALUE n
FROM ChicagoCrimes n LIMIT 1;
```


```
{
 "ChicagoCrimes": {
  "id": "d85704ce-ac9f-65f0-d6d6-2deb29206d07",
  "g": {
   "type": "Point",
   "coordinates": [
    -87.883611316,
    41.980826277
   ],
   "crs": {
    "type": "name",
    "properties": {
     "name": "EPSG:4326"
    }
   }
  },
  "ID": "9805746",
  "Case Number": "HX442584",
  "Primary Type": "MOTOR VEHICLE THEFT",
  "Description": "AUTOMOBILE",
  "Location Description": "AIRPORT VENDING
ESTABLISHMENT",
  "Arrest": "false",
  "Domestic": "false",
  ...
  "Updated On": "02/10/2018 03:50:01 PM"
 }
}
```

External Dataset

```
// External datasets
CREATE TYPE RoadsTypeExternal IF NOT EXISTS AS {
  g: geometry,
  FULLNAME: String
};

CREATE EXTERNAL DATASET RoadsExternal(RoadsTypeExternal)
USING localfs
(("path"="127.0.0.1:///home/admin/bossdata/roads.json"),
("format"="adm"));

SELECT VALUE r
FROM RoadsExternal r
LIMIT 1;
```

Good for in-situ processing of static data (Similar to Hadoop and Spark)

Example Query 12

Problem: List the Roads that cross each other.

We do a self join

```
SELECT r1.FULLNAME AS r1_name, r2.FULLNAME AS r2_name
FROM Roads r1, Roads r2
WHERE st_crosses(r1.g, r2.g) AND r2.id > r1.id
ORDER BY r1.FULLNAME;
```

We check if two roads are crossing each other and apply a duplicate avoidance.

Multi-dataset and nested queries - Query 13

Problem: Find the average number of neighbors for Zip Code boundaries.

- Do a self join on `ZipCodeBoundaries`
- Check if Geometries touch or not with `st_touches()`
- Use GROUP BY `ZCTA5CE10 - the zip code value`
- Use COUNT()
- Use AVG()

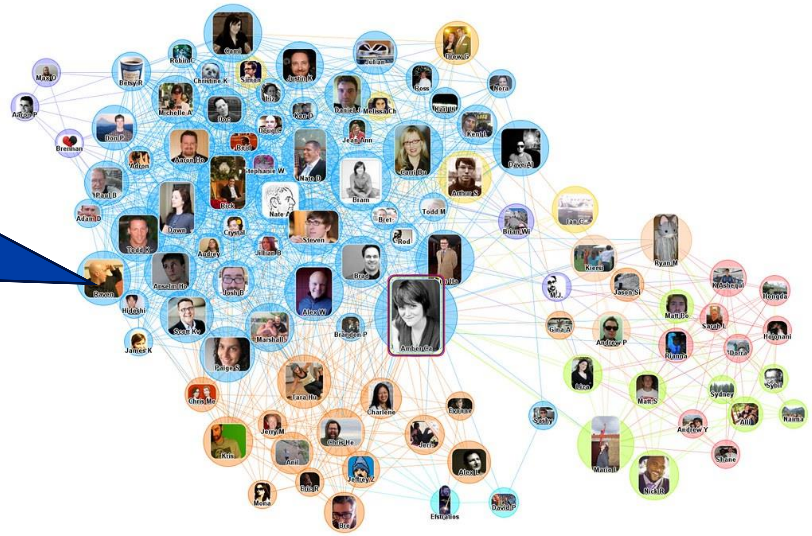
```
SELECT AVG(c) FROM (
  SELECT COUNT(1) as c
    FROM ZipCodeBoundaries AS z1, ZipCodeBoundaries AS z2
    WHERE st_touches(z1.g, z2.g)
    GROUP BY z1.ZCTA5CE10
) as t;
```

Graph Processing

Big Graphs

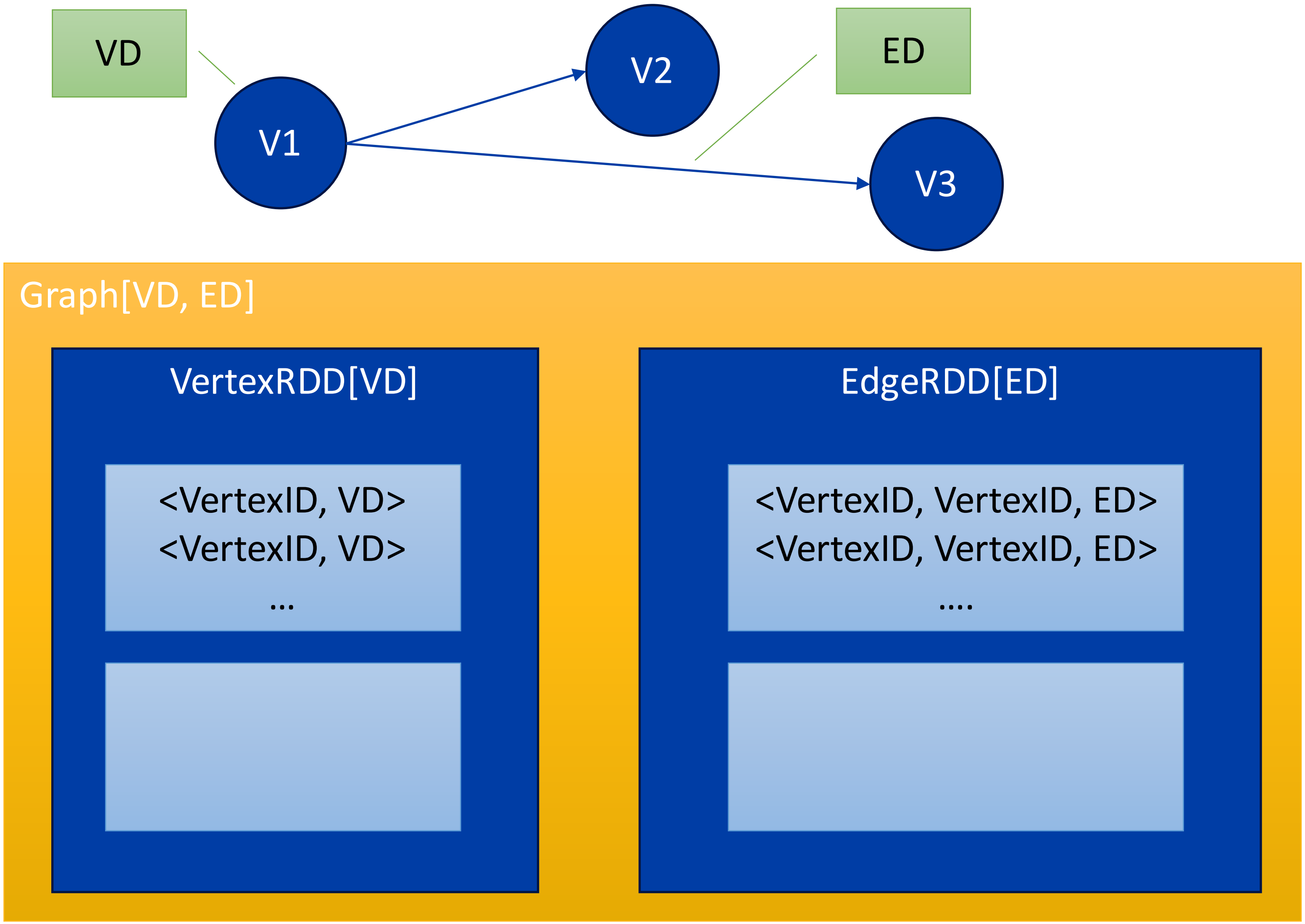
- Graphs with hundreds of millions of vertices and billions of edges

1.7 billion users
100s of billions of connections



- The graph is too big to fit on one machine

Property Graph Model

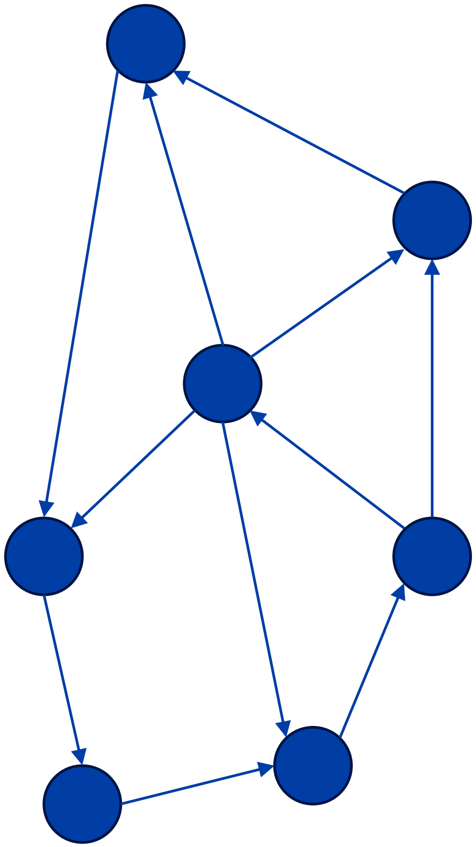


Examples of Graphs

- Knowledge Graph
  - Graph[String, String]
  - VD: String – entity names
  - ED: String – relationship
- Road Network
  - VD: (longitude, Latitude)
  - ED: (StreetName, SpeedLimit, …)

Iterative Graph Processing

- Each vertex send a message along outgoing edges
- Each vertex updates its vertex data by aggregating incoming messages
- Repeat until a stopping condition
- Pregel API



CELEBRATING 30 YEARS
UCR Marlan and Rosemary Bourns College of Engineering

Vector DBMS

The Rise of AI

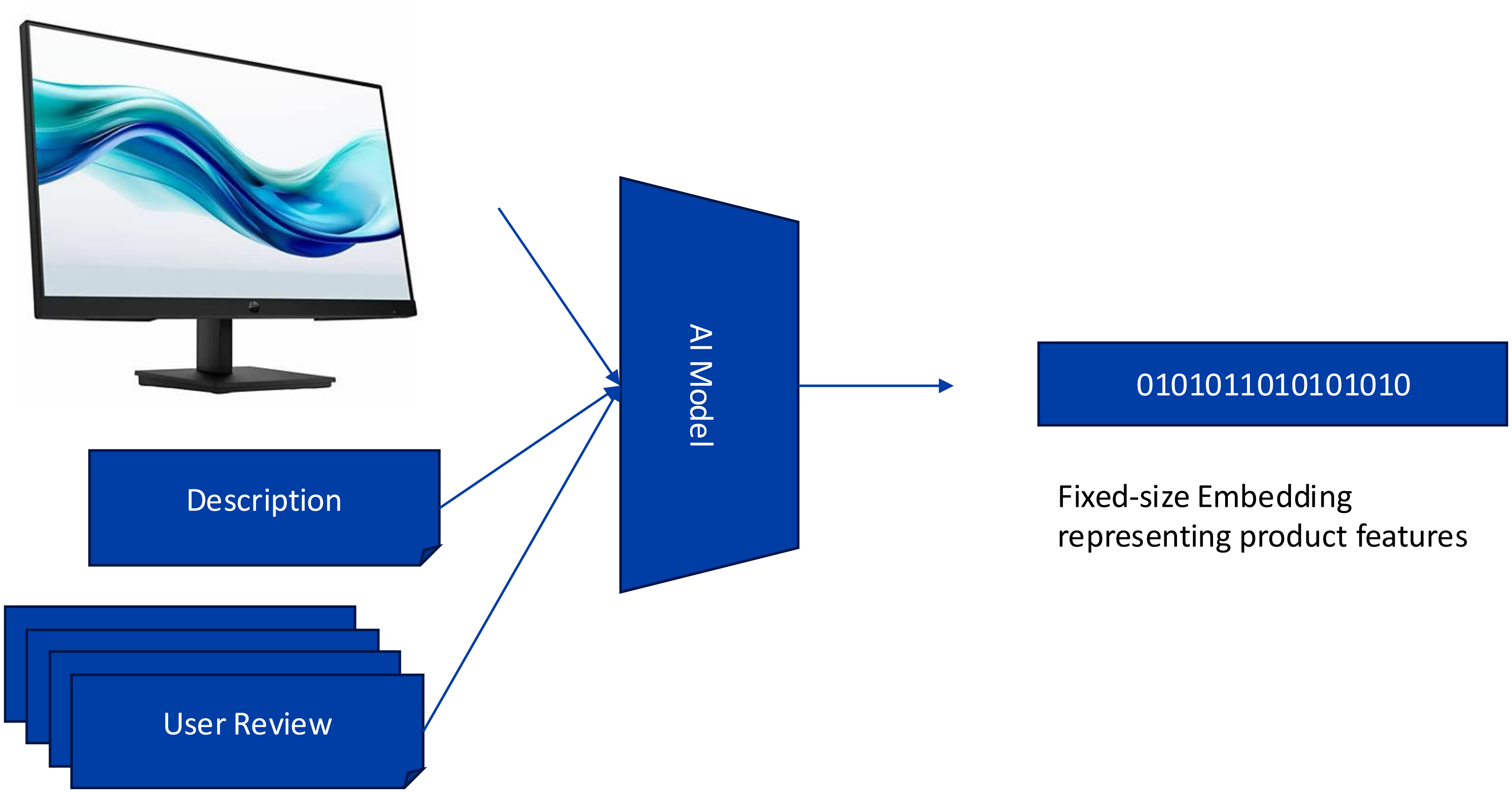
- Unstructured queries are becoming more prevalent with the rise of Large Language Models (LLMs)
  - "Find the five cheapest 24" computer monitors that have at least 4.5 star

```sql
SELECT id, name, size_inches, price, rating
FROM monitors
WHERE size_inches = 24 AND rating >= 4.5
ORDER BY price ASC
LIMIT 5;
```

More Interesting Unstructured Queries

- In a database of products with pictures, description, and user reviews
  - "Find highly rated products that look like this but has a white color and is preferred by graphic designers"

AI Embedding of Records



Semantic Search

“Find a highly rated product that looks like this but has a white color and is preferred by graphic designers”

AI Model

0101101011100010

Query Embedding

Nearest Neighbor Search

0101111011100011
1101101011110010
1101101001100010
0111101011100011
1101101011100010
0111101001110111

RAG Question Answering

- RAG: Retrieval Augmented Generation
- Given a query:
  - Create a query embedding
  - Run an approximate nearest neighbor (ANN) query to find the most relevant documents
  - With an appropriate prompt, feed the documents through a generative AI model to get the final answer

VDBMS