
# A brief overview of Apache Spark

Prof. Pietro Michiardi

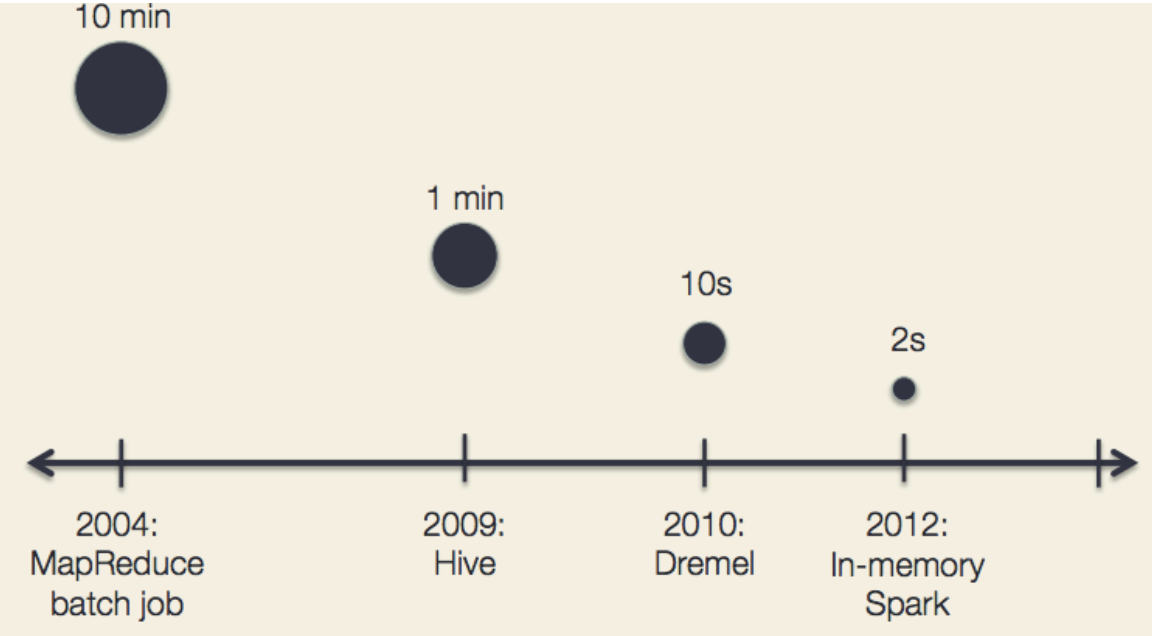
Data Science Department

EURECOM

Sophia Antipolis

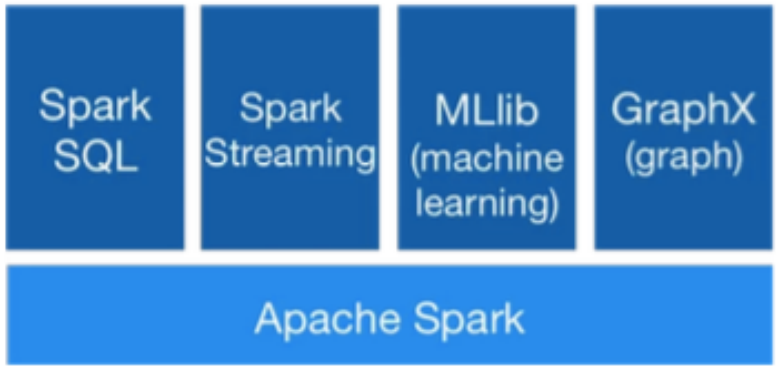
# Trends in the Big Data landscape

- **Initially**: focus on **batch** processing
- **Next**: **high-level** languages to compensate strict, low-level programming model
- **Now**: **low latency**

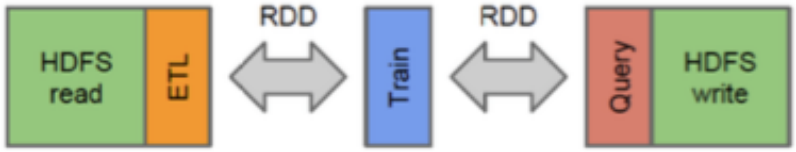


# Apache Spark: Objectives

- **Project goals**
  - Heterogeneous workloads
  - Low latency: sub-second
  - Fault tolerance
  - Simplicity




- **System/Framework point of view**
  - Unified pipeline
  - Simplified data flow
  - Faster processing speed




- **Data abstraction point of view**
  - New fundamental abstraction RDD
  - Easy to extend with new operators
  - More descriptive computing model

# Logistic Regression with Gradient Descent

$$\mathcal{L}(w; x, y) = \sum_{i=1}^{n} F(w; x_i, y_i)$$

**Loss function to minimize**

$$g(w; x_i, y_i) = \nabla F(w; x_i, y_i)|_w$$

**Gradient w.r.t. parameters**

$$w \leftarrow w - \alpha \cdot \sum_{i=1}^{n} g(w; x_i, y_i)$$

**Gradient descent update**

```
val points = spark.textFile(...).map(parsePoint).cache()
var w = Vector.zeros(d)
for (i <- 1 to numIterations) {
  val gradient = points.map { p =>
    (1 / (1 + exp(-p.y * w.dot(p.x)) - 1) * p.y * p.x
  ).reduce(_ + _)
  w -= alpha * gradient
}
```

# Apache Spark: a Data Scientist perspective

- **Development cycle**
  - Notebooks to **design** and **debug**
  - Standalone to **tune** (both algos and system)

- **Main issues encountered regularly**
  - Caching: what?, when?, when to un-persist?
  - Memory issues: jobs crash
  - GC issues: jobs become extremely slow
  - Capacity planning, tuning: art or science?
  - Careful in choosing your API: scala, python, R …

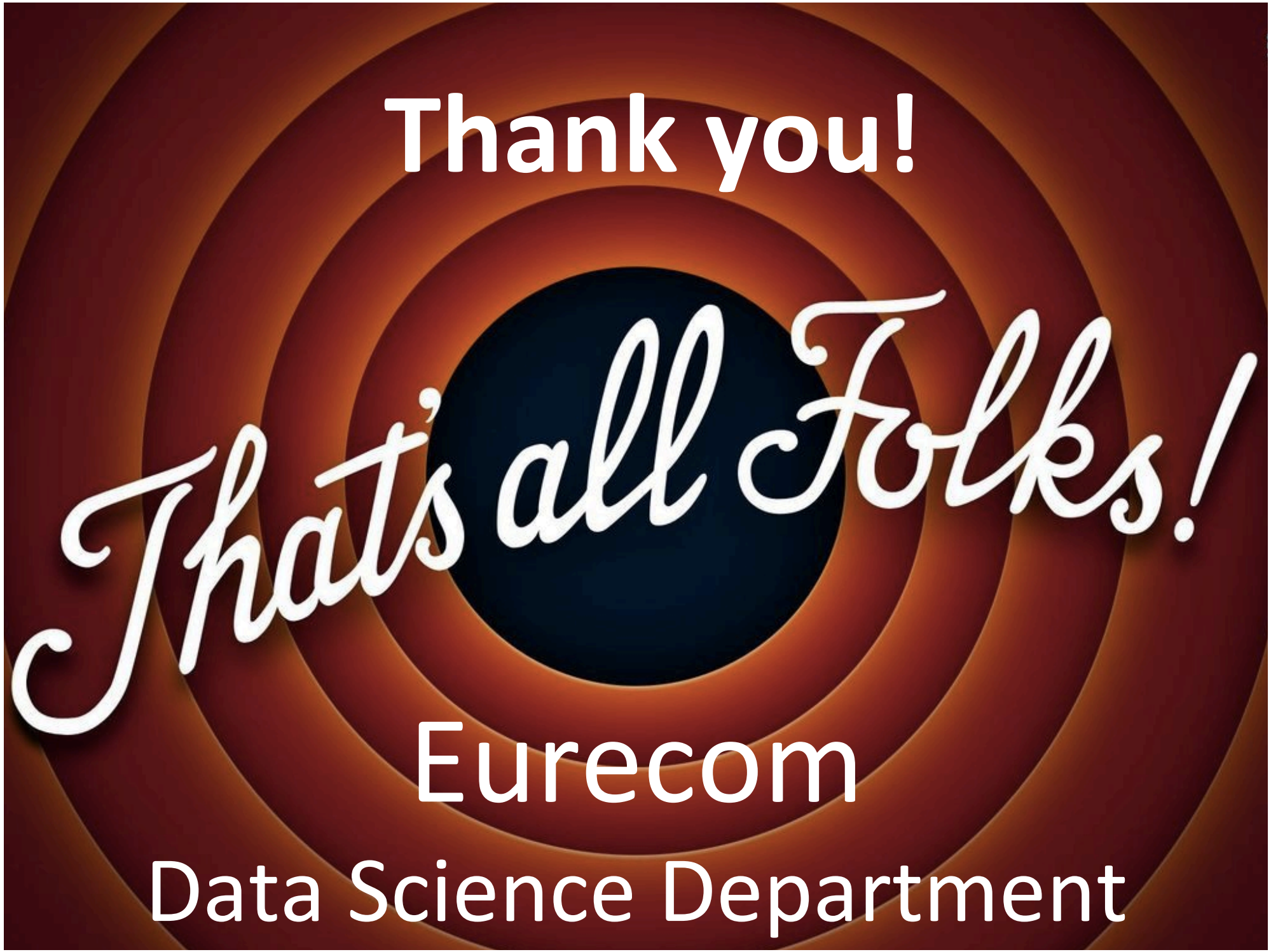
# Apache Spark: the future

- **Apache Spark 2.0**
  - **Tungsten**: memory + CPU optimizations
  - **Catalyst**: a Spark SQL query optimizer as in DBMS
  - DataFrames / Datasets: high-level data structure on RDD
  - Machine Learning **Pipes**

- **Personal viewpoint**
  - IMHO Spark is heading to be a parallel DB-like product
  - Ease of adoption: Jupyter Notebooks, databricks cloud
  - Consolidation is good!
  - No “new” programming models
    - Apache Flink, Google DataFlow, Google TensorFlow, MS Naiad …

# Conclusion

- **Databricks is bringing spark to the enterprise**
- **Spark is still a hard beast to deal with**
  - From “hello world” to sophisticated production pipelines
- **Advice for Data Scientist wannabes**
  - Understand the fundamentals in machine learning (math)
  - Use python notebooks
    - Sci-kit learn if your data is small
    - Spark when you have scalability problems
  - Spend time in data preparation and information extraction


Thank you!
That's all Folks!
Eurecom
Data Science Department