# Analyzing the Graph-Processing Pipeline: A comparative study of GraphLab and GraphX

An open source project study

Presented by Niko Stahl for R212

# Context

- **GraphLab** (execution engine: Powergraph) is exclusively built for graph processing.
- **GraphX** is built on top of Spark.

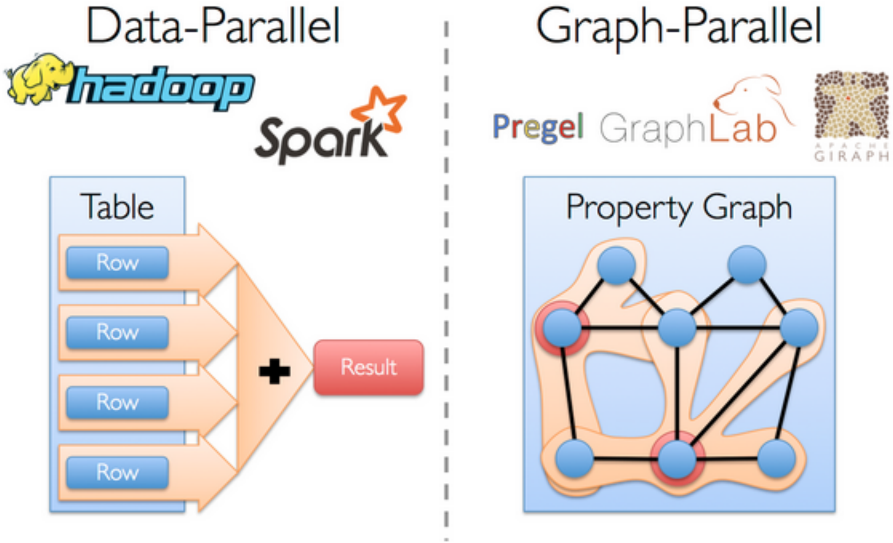
# Quick Intro: GraphX and Spark

What makes it competitive?

- Spark facilitates in-memory computation on clusters.
- The main abstraction:
  **RDDs (Resilient Distributed Datasets)**
- RDDs maintain fault tolerance
- The caching of RDDs can greatly speed-up algorithms that exhibit data **reuse** (e.g. PageRank)

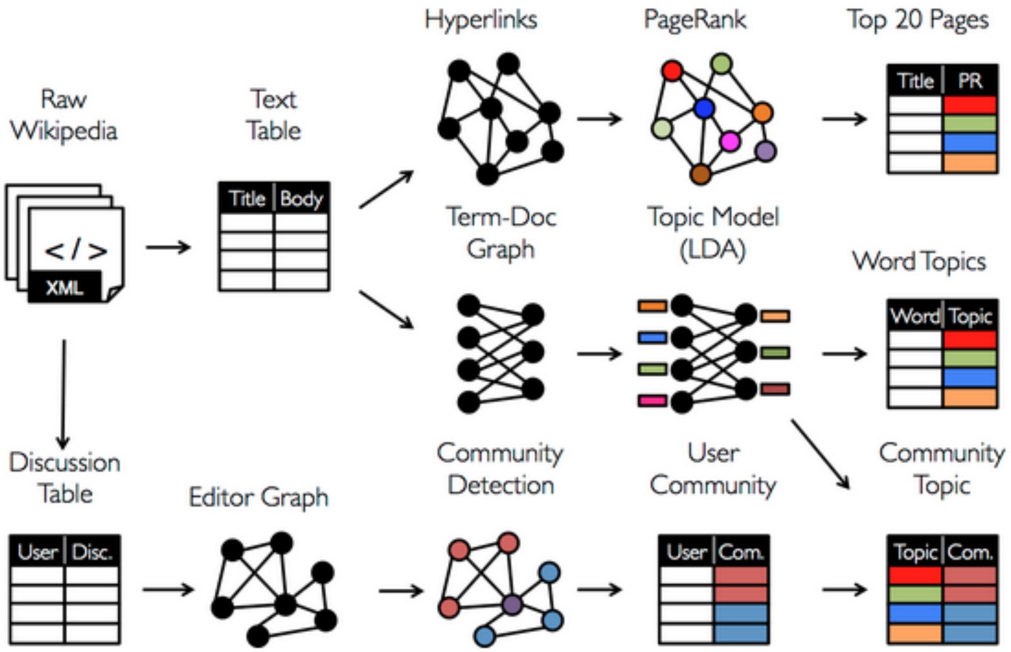
# Context

- GraphX combines the advantages of data-parallel and graph-parallel systems.



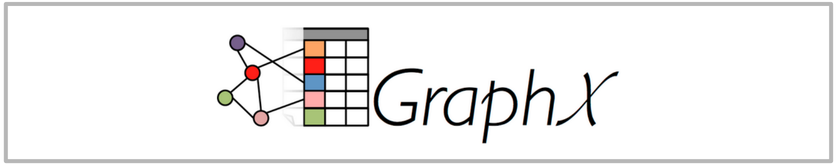
# Why is it useful to combine data-parallel and graph-parallel features?

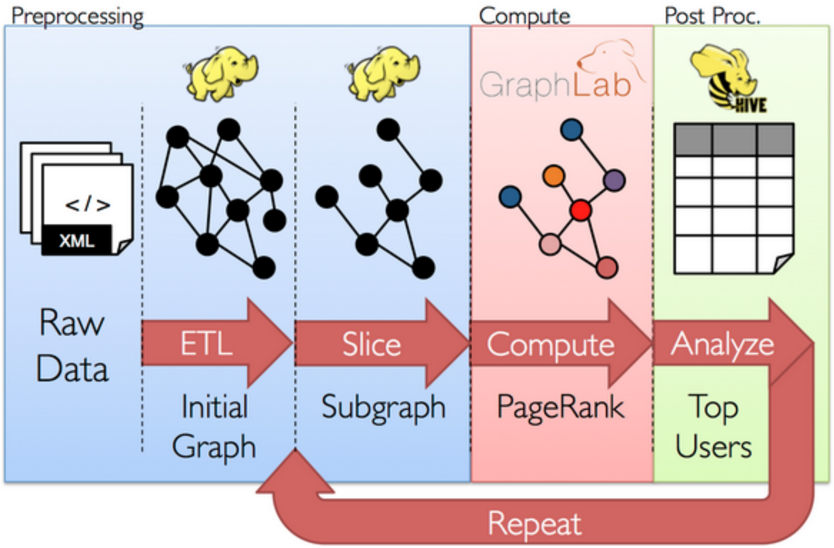
A typical graph-processing pipeline requires moving between different views of the same data.



http://spark.apache.org/docs/0.9.0/graphx-programming-guide.html

# Context Switching: GraphX preferred





http://spark.apache.org/docs/0.9.0/graphx-programming-guide.html

# Performance: GraphLab preferred

## Xin et al., 2013: GraphX: A Resilient Distributed Graph System on Spark

16 node Amazon EC2 cluster

Each node 8 virtual cores

68GB memory

Graph: 4.8M vertices, 69M edges

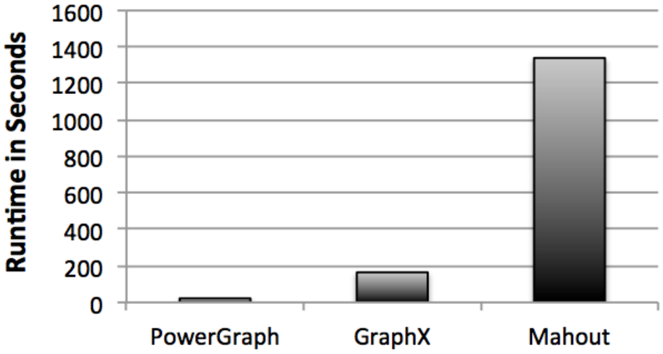


Figure 4: **PageRank Runtime Comparison** between GraphX, Mahout/Hadoop, and PowerGraph. The reported runtime includes the time to load the graph from HDFS and then run 10 iterations of PageRank.

# Project Motivation

***“We believe that the loss in performance may, in many cases, be ameliorated by the gains in productivity achieved by the GraphX system.”*** - Xin et al., 2013

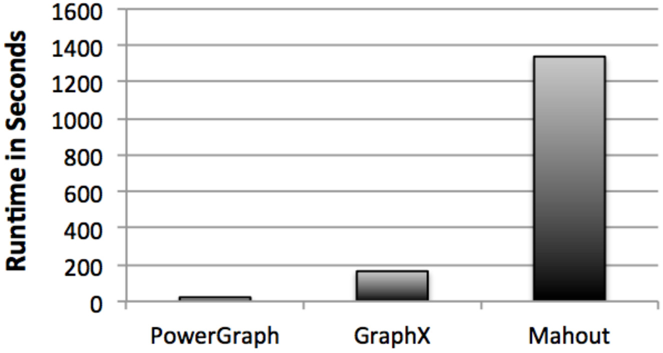


Figure 4: **PageRank Runtime Comparison** between GraphX, Mahout/Hadoop, and PowerGraph. The reported runtime includes the time to load the graph from HDFS and then run 10 iterations of PageRank.

# Project Significance

- GraphLab released **GraphLab Create** earlier this year
- Goal of the project is to introduce a **tabular data structure** (SFrame) to GraphLab
- SFrame are similar to R/pandas data frames but stored on disk.
- To the best of my knowledge, there are no direct comparisons between GraphLab Create and GraphX.

# Project Aim - In Detail

- Compare the efficiency and usability of GraphLab Create vs. GraphX in a **realistic scenario**.
- The pipeline I will evaluate:

1. **transform** (Filter pages of a certain language)
2. **process** (PageRank)
3. **summarize** (top k most influential pages)

# Project Evaluation

- Experiments will take place on an Amazon EC2 cluster
- Each stage will be evaluated according to:
    1. Execution Time
    2. Programming effort (lines of code, flexibility of API)

# Expected Outcome

| stage | performance | programming effort |
|---|---|---|
| 1. transform | GraphX (?) | ? |
| 2. process | GraphLab | ? |
| 3. summarize | GraphX (?) | ? |

# Project Challenges

- How objective is a comparison on Amazon EC2?
  -> Every time you launch a cluster you get different machines.
- How do you objectively evaluate programming effort?
  -> Lines of code is contrived. This will be a subjective evaluation.

# Project Status

- I have launched GraphX on AmazonEC2 and have run stand-alone Scala applications with GraphX.
- Next Steps:
  1. Setup preliminary GraphX experiments
  2. Setup preliminary GraphLab Create experiments
  3. Evaluate how comparable each stage is
  4. Tune experiments and run repeatedly on Amazon EC2 to get statistics