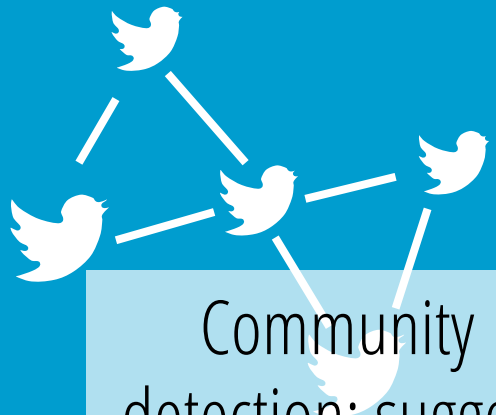
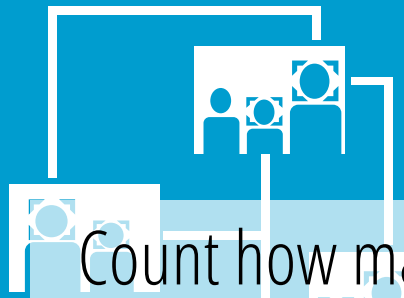
Why do we need graph processing?



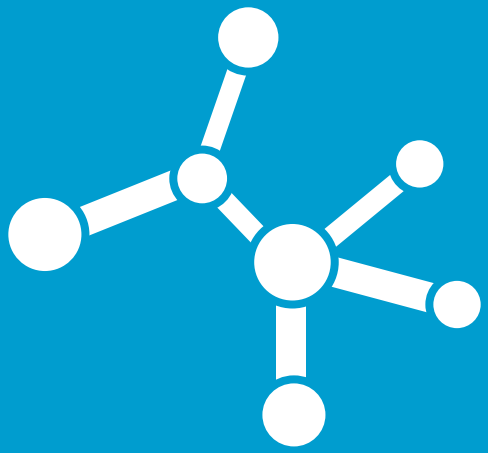
Community detection: suggest followers?

Determine what products people will like



Count how many people are in different communities (polling?)

Graphs are Everywhere

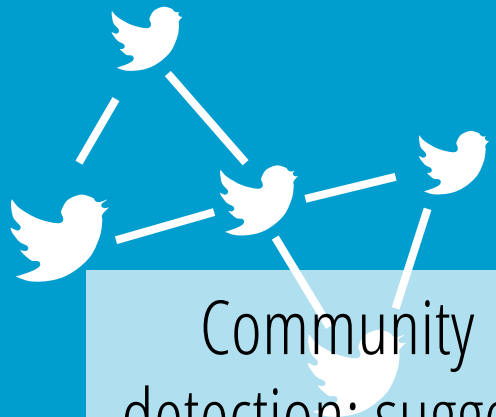


Group similar articles

Why do we need graph processing?

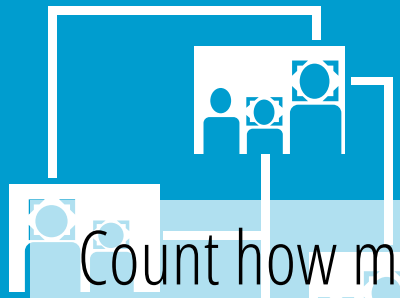
- Collaborative Filtering
  - Alternating Least Squares
  - Stochastic Gradient Descent
  - Tensor Factorization
- Structured Prediction
  - Loopy Belief Propagation
  - Max-Product Linear Programs
  - Gibbs Sampling
- Semi-supervised ML
  - Graph SSL
  - CoEM
- Community Detection
  - Triangle-Counting
  - K-core Decomposition
  - K-Truss
- Graph Analytics
  - PageRank
  - Personalized PageRank
  - Shortest Path
  - Graph Coloring

Many of these can be expressed as matrix problems



Community detection: suggest followers?

Determine what products people will like

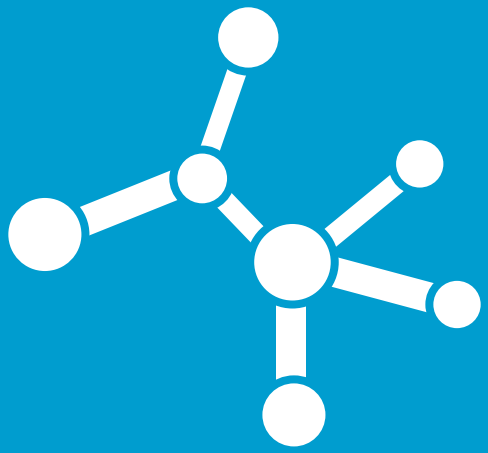


Count how many people are in different communities (polling?)

Graphs are Everywhere

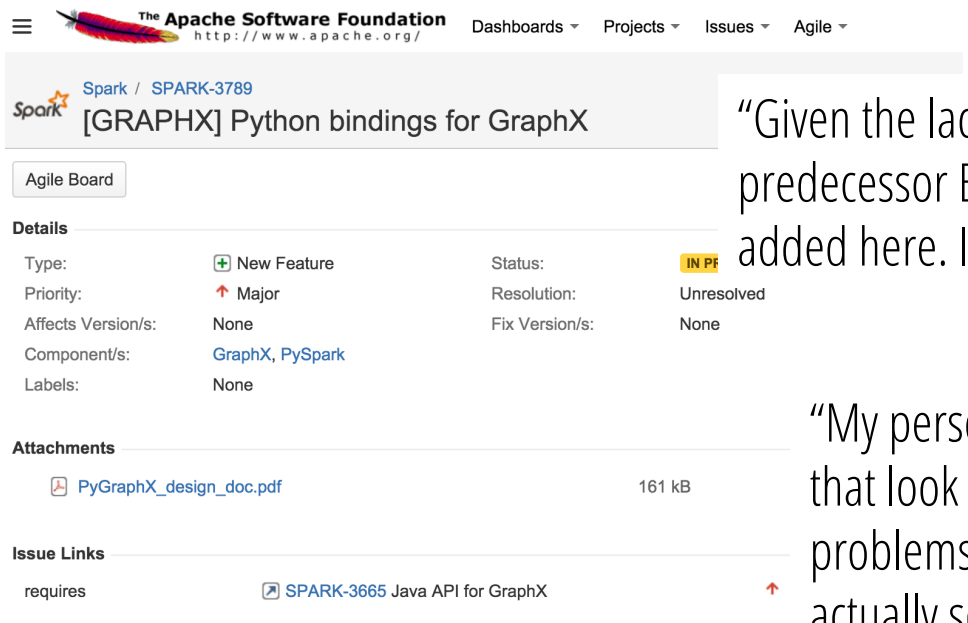
Collaborative filtering

?



Clustering

Group similar articles



"Given the lack of activity on GraphX and its defunct predecessor Bagel, I doubt anything significant will be added here. I'd almost just close this."

"My personal bias is that most real-world problems that look like they'd be cool to solve as graph problems, aren't graph problems or aren't great to actually solve that way"

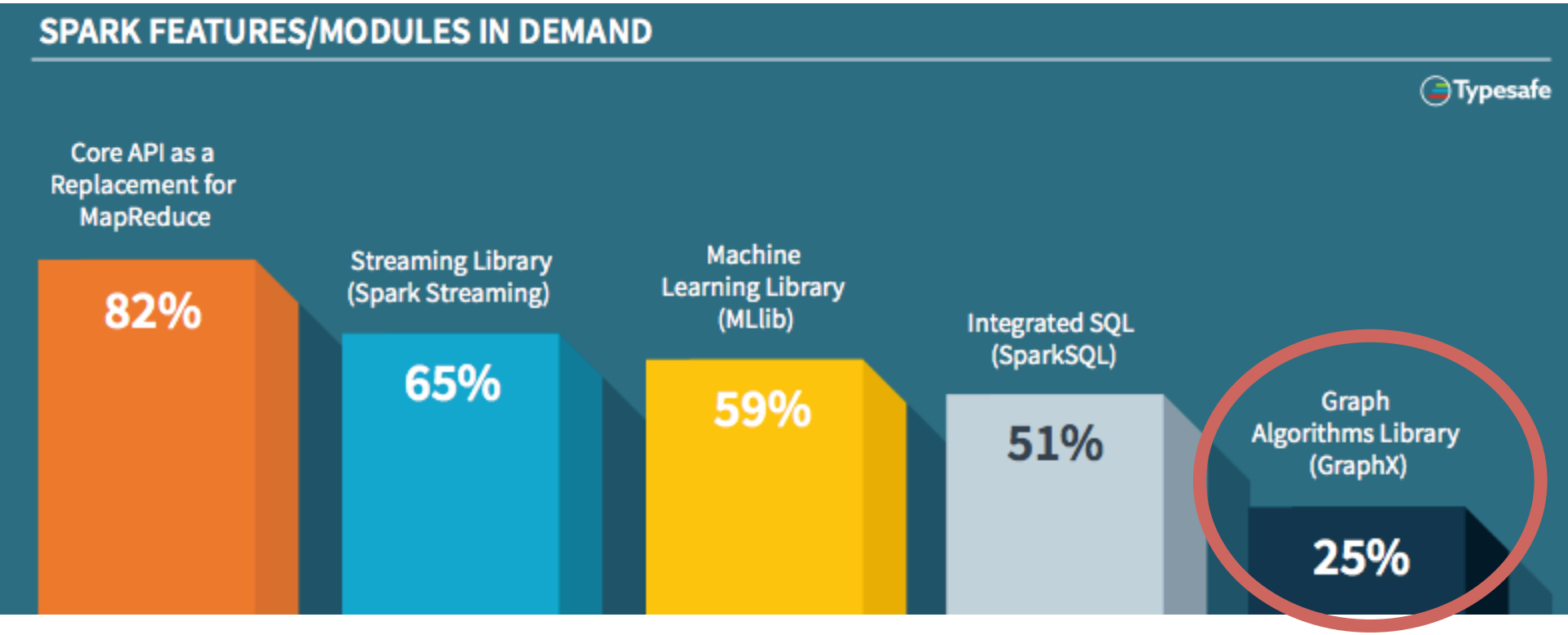
"**The entire world of ecommerce on the internet is driven by graph analytics** (page rank, suggestions, also viewed, etc. etc.) While arcane to some, is a very important and growing field of computer science and web analytics. ESPECIALLY where big data is concerned."

"FWIW virtually none of our customers use GraphX, and we interact with a pretty good cross section of Big Companies. Many of the useful functions you identify are not solved as graph problems in my experience, even if they could be (e.g. recommenders, also viewed)."

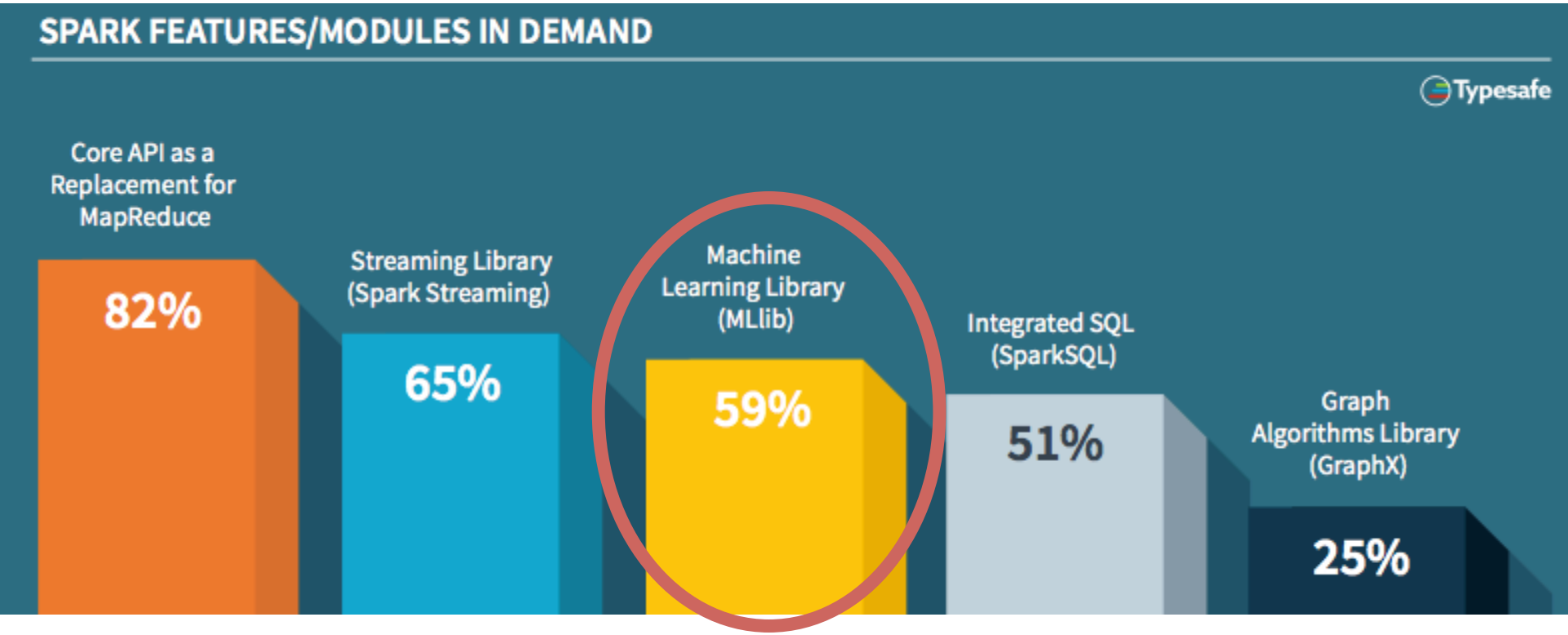
Can we infer from the comments on this ticket that GraphX will be discontinued and no longer supported by the Spark Community? I see the makings of rumors already...

Which of the following Spark features or modules are most likely to solve your big data challenges?

(Typesafe + Databricks + Dzone surveyed 2136 people)



Some Mllib algorithms use GraphX: Power Iteration Clustering, LDA

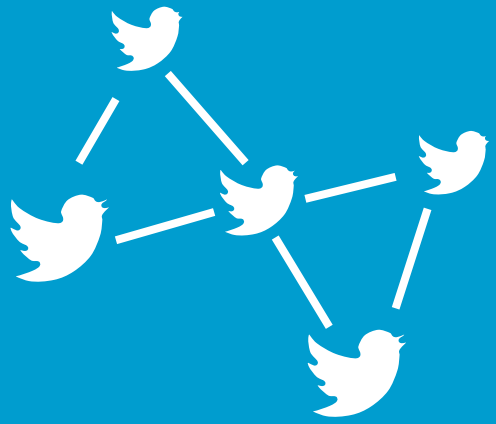


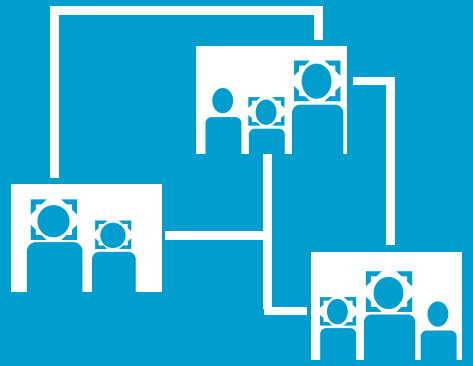
Trends

Why do we need **distributed** graph processing?

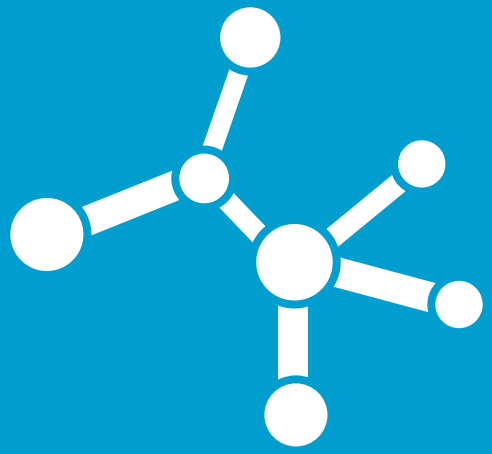
- Graphs used in GraphX paper have billions of edges
  - Twitter: 40m users, 1.4billion links

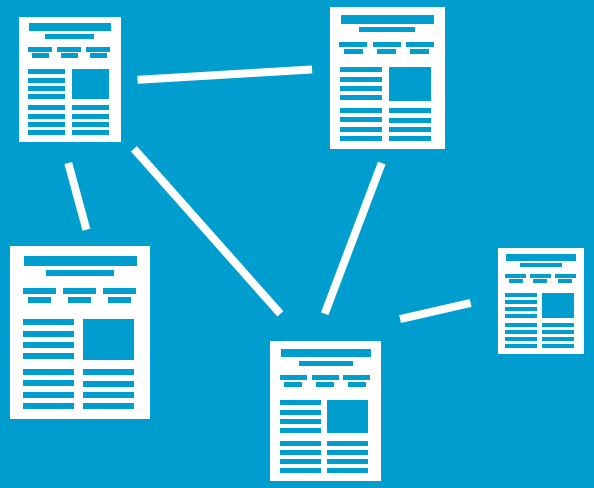
- Frank McSherry's laptop can process them faster than GraphX





Big | Billions of Edges
Rich Metadata



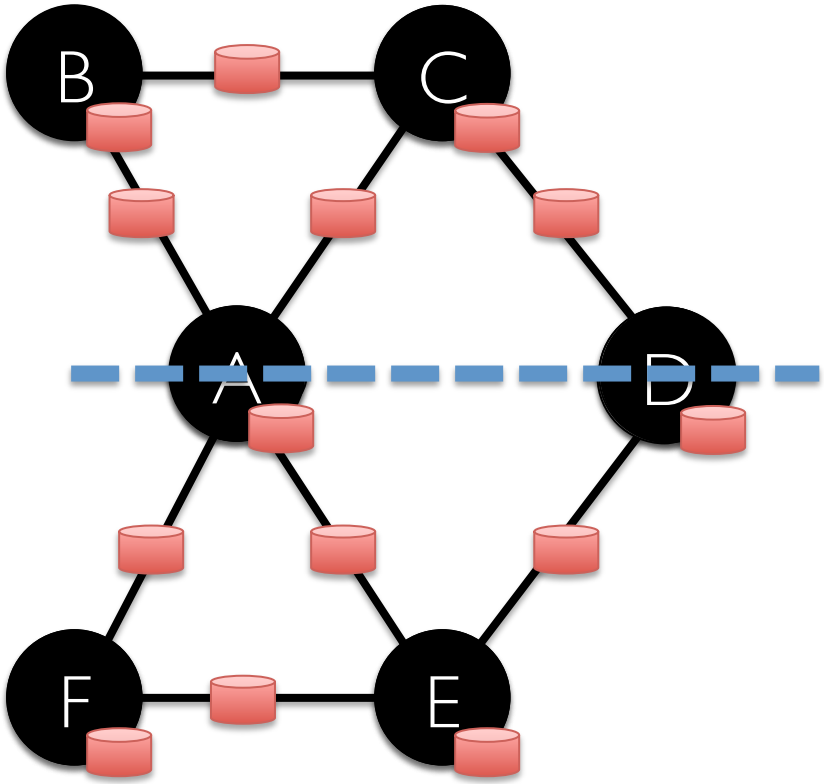


Why do we need **distributed** graph processing?

- To process graphs with lots of metadata
  - Is the metadata needed for the graph problem?
- Because it's convenient to incorporate in a single system
  - Include fast single-machine implementation (as fallback) in Spark?

What's hard about distributed graph processing?

- How do you represent the graph?
- How do you distribute the graph over the machine?
  - Some parts of the graph need to be duplicated
  - Existing systems: specialized representation

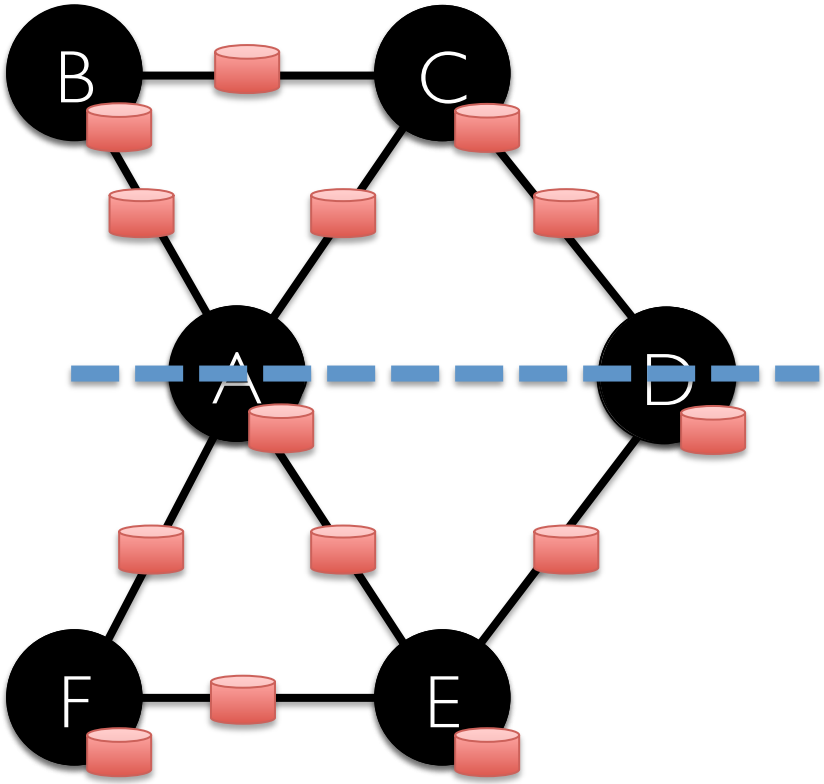


GraphX: can we use a general-purpose system?

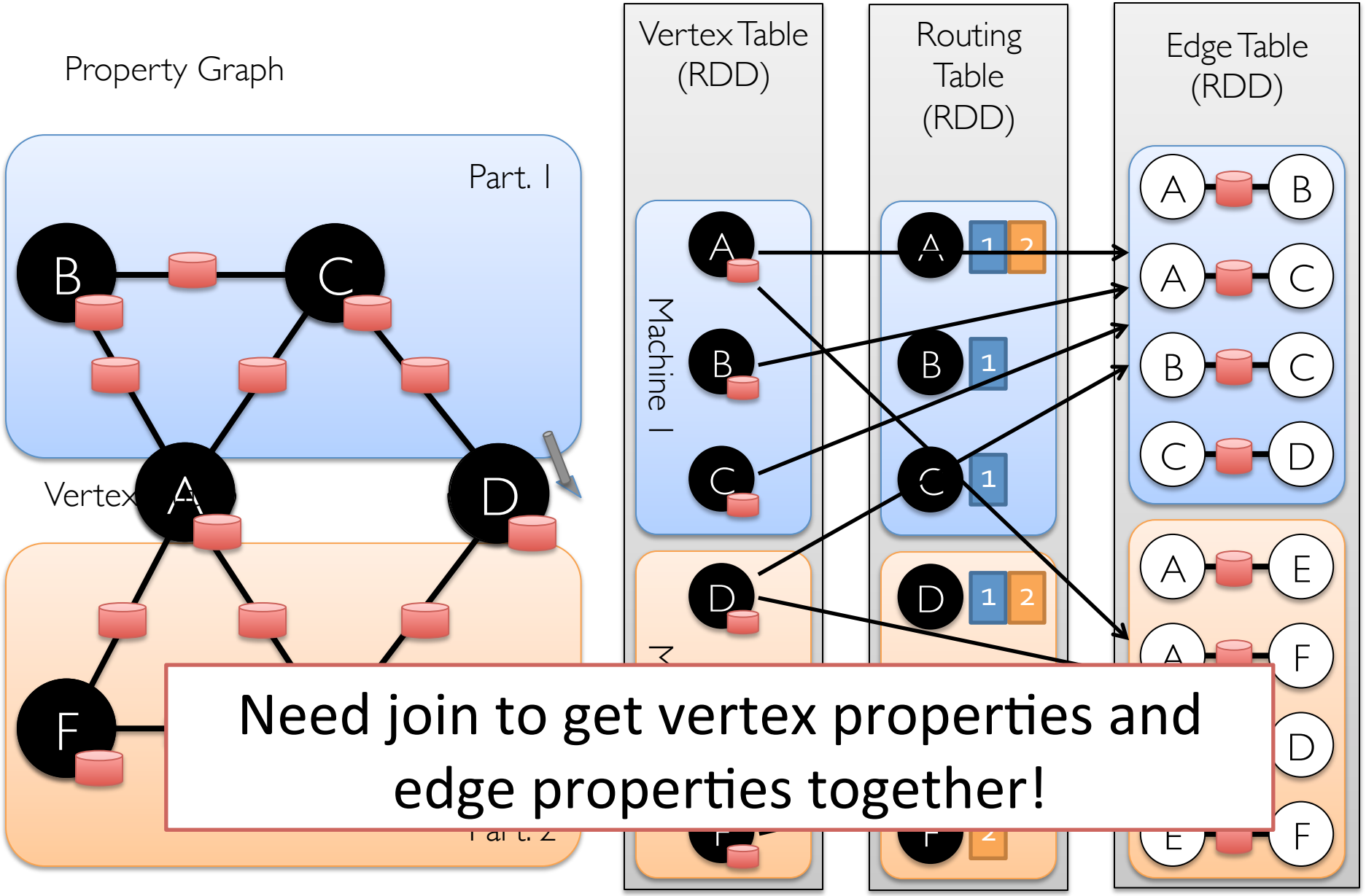
- Graph computation often part of a bigger pipeline
- Why now?
  - Frameworks support in-memory processing (kind of)
  - Allow fine-grained control over data partitioning

Fundamental challenge: data representation

Dataflow systems expect a single, partitioned dataset



Encoding Property Graphs as Tables



What changed?

Graph Processing Systems

- Pregel: synchronous steps
- GraphLab: asynchronous steps
- PowerGraph: better placement / representation
- GraphX: built on general purpose system, synchronous

Takeaway: Spark as a building block

- Gave control over data storage (memory / disk)
- Gave control over data partitioning
- Doesn't support asynchrony

Existing paper says: network doesn't matter for GraphX

I optimized the CPU time of PageRank

Now the network matters more

Why are Frank McSherry's things so much faster than GraphX?

- His laptop was faster
- Timely dataflow was much faster
- Cost of generality?
  - With simple types, GraphX spends much of its time boxing/unboxing primitive types
  - Serialization is not optimized
  - Spark is in the process of improving this