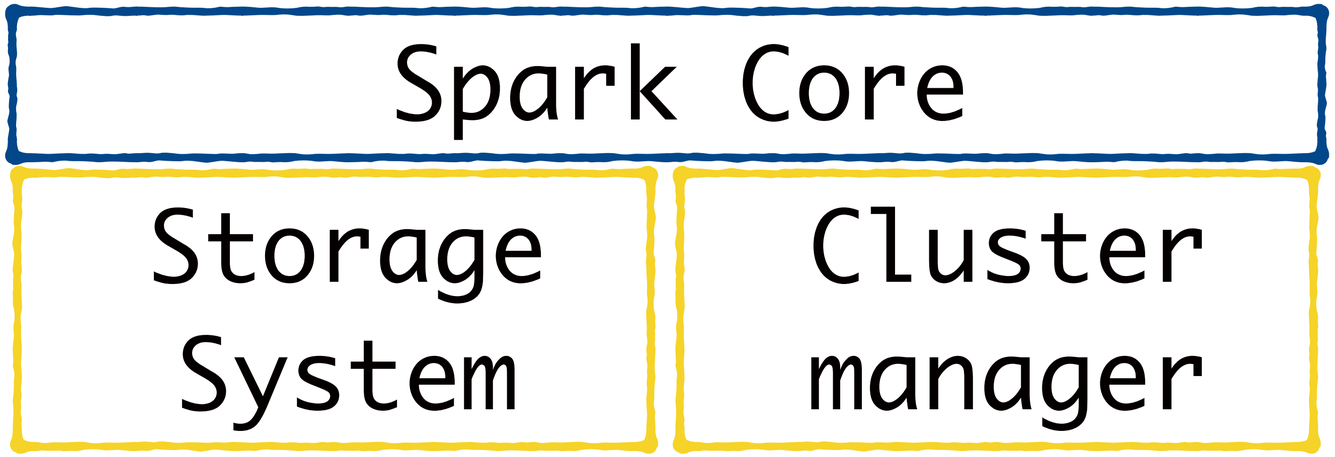
GRAPHX

Recap

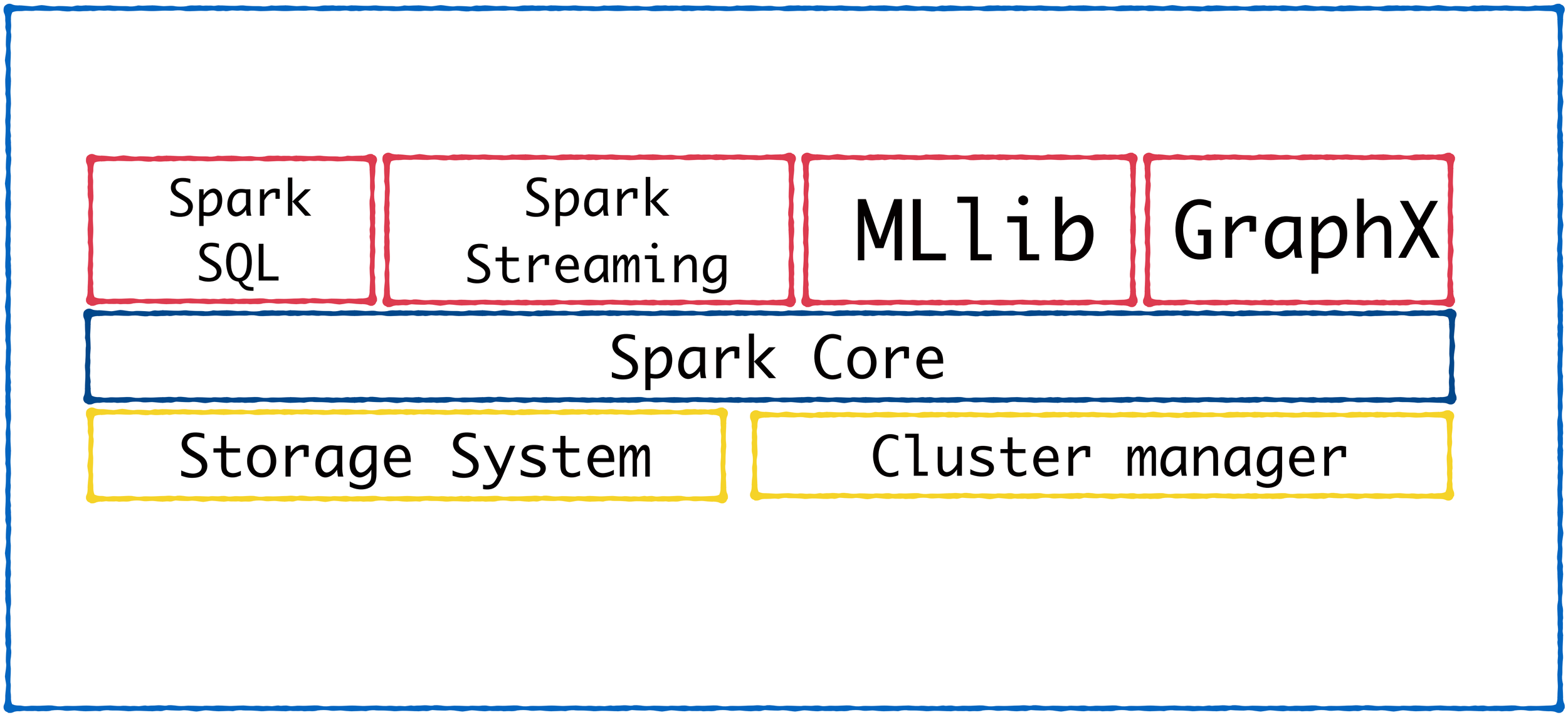
APACHE SPARK

Spark comes with some additional packages that make it truly general - purpose



Recap

APACHE SPARK



Recap

APACHE SPARK

Spark SQL | Spark Streaming | MLlib | GraphX

Spark Core

Storage System | Cluster Manager

GraphX is a library for graph algorithms

Recap

APACHE SPARK

Spark SQL | Spark Streaming | MLlib | GraphX

Spark Core

Storage Systems | Cluster Managers

Many interesting datasets can be represented as graphs

Recap

APACHE SPARK

Spark SQL | Spark Streaming | MLlib | GraphX

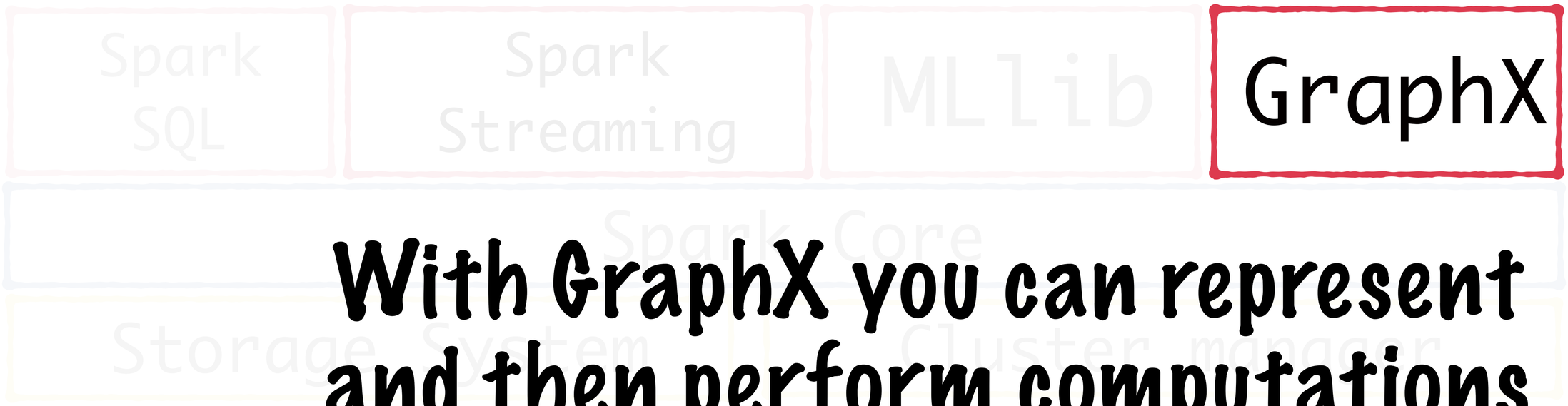
Spark Core

Storage System | Cluster Manager

Social networks, linked webpages etc

Recap

APACHE SPARK



With GraphX you can represent and then perform computations across these datasets

Recap

APACHE SPARK

Just like with MLlib, GraphX abstracts the programmer from having to implement Graph algorithms

GraphX

Recap

APACHE SPARK

Because of RDDs and in-memory computation, GraphX on Spark gives you better performance than other computing frameworks (like MapReduce)

GraphX

GraphX

GraphX is only available for Scala

For Python, a separate module called GraphFrames is available

GraphFrames

GraphX

This module has most of the functionality that GraphX provides

It is slated to be integrated into Spark and with GraphX in the near future

GraphFrames

GraphX

Let's use GraphFrames to explore an interesting dataset

Marvel Social Network

Marvel Social Network

GraphX

Marvel Social Network

GraphX

The Marvel Comic Book Universe has thousands of characters

Marvel Social Network

GraphX

The relationships between these characters could be explored using a graph

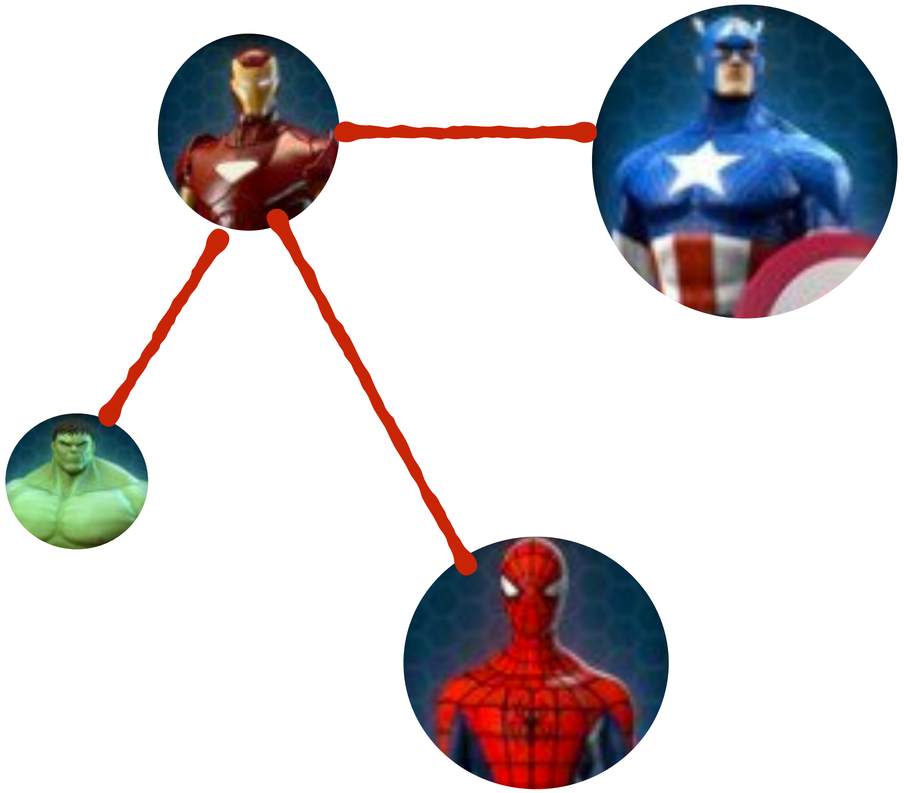
Marvel Social Network

GraphX

The graph would have

1. Characters as vertices

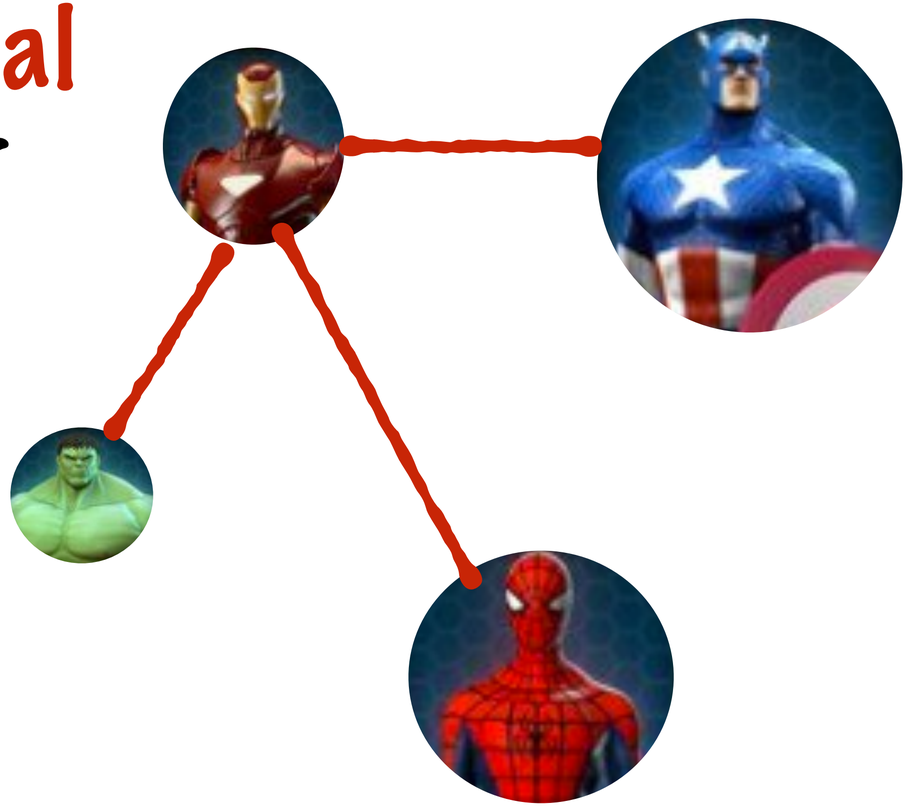
2. An edge representing if 2 characters appear together in a book



Marvel Social Network

GraphX

1. Size of the vertices is proportional to number of times the character appears in any comic book

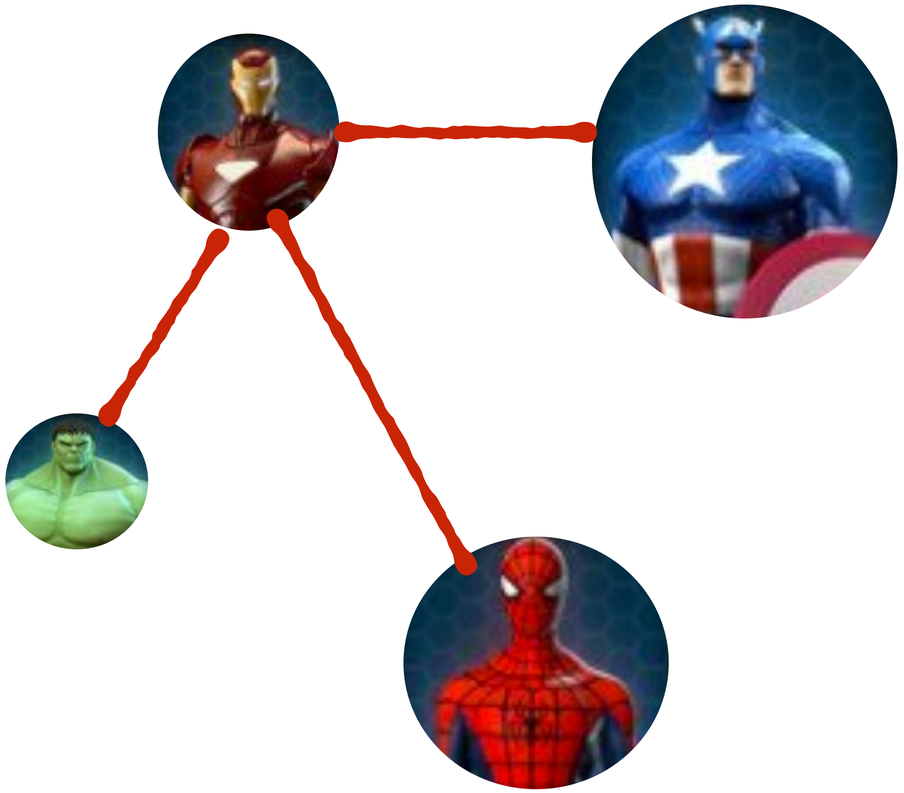


2. Edge weight is proportional to the number of times the characters appear together

Marvel Social Network

GraphX

Such graphs are called Co-occurence networks



Marvel Social Network

GraphX

The dataset for this exercise was constructed using raw data provided by

Author ©:Joe Miro

who did a study on

Social characteristics of the Marvel Universe

Marvel Social Network

GraphX

The dataset has

A Vertices file

Character id,
Name,
Times the character appears in any book

An Edges file

Character 1,
Character 2,
Times they appear together

Marvel Social Network

GraphX

To represent a graph in Spark, you need

A Vertices DataFrame

An Edges DataFrame

Marvel Social Network

GraphX

A Vertices DataFrame

An Edges DataFrame

Recall: A DataFrame is like an in-memory table

Marvel Social Network

GraphX

A Vertices DataFrame

An Edges DataFrame

This DataFrame must have an "id" field

Marvel Social Network

GraphX

A Vertices DataFrame "id" field

An Edges DataFrame

It can have other fields as well, but these are left to the user's discretion

Marvel Social Network

GraphX

A Vertices DataFrame "id" field

An Edges DataFrame

The other fields may represent attributes of the vertex Ex: Name

Marvel Social Network

GraphX

A Vertices DataFrame "id" field

An Edges DataFrame

This must have an "src" field and a "dst" field

Marvel Social Network

GraphX

A Vertices DataFrame "id" field

An Edges DataFrame

It can have other fields corresponding to edge attributes Ex: Edge weight

Marvel Social Network

GraphX

```
from pyspark.sql import SQLContext, Row
sqlC=SQLContext(sc)
```

We need an SQLContext to set up DataFrames

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

Here is the edges DataFrame

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

We've seen an example where we used sqlC.read to create a DataFrame

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

The read method is used when you are reading data directly from file

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

This is another way to create a DataFrame

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

We need to give this method an RDD

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

We need to give this method an RDD

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

And a schema

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

We know that our edges DataFrame must have "src" and "dst" fields

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

Using createDataFrame we can explicitly specify the schema for the DataFrame

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

In addition to the mandatory fields, we also specify a weight

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

This is the count of the number of times the 2 characters appear together

Marvel Social Network

GraphX

```
edges=sqlC.createDataFrame\
    (sc.textFile(edgesPath)\
    .map(lambda x:x.split(","))\
    .map(lambda x:[int(y) for y in x])\
    ,["src","dst","wt"])
```

Load the data from file into an RDD, parse the rows and give it to createDataFrame

Marvel Social Network

GraphX

```
vertices=sqlC.createDataFrame\
    (sc.textFile(verticesPath)\
    .map(lambda x:x.split("|"))\
    .map(lambda x:[int(x[0]),x[1],int(x[2])])\
    ,["id","name","wt"])
```

Similarly, we can set up a vertices DataFrame with a schema

Marvel Social Network

GraphX

```
vertices=sqlC.createDataFrame\
     (sc.textFile(verticesPath)\
     .map(lambda x:x.split("|"))\
     .map(lambda x:[int(x[0]),x[1],int(x[2])])\
     ,["id","name","wt"])
```

The vertices DataFrame needs an "id" field

Marvel Social Network

GraphX

```
vertices=sqlC.createDataFrame\
    (sc.textFile(verticesPath)\
    .map(lambda x:x.split("|"))\
    .map(lambda x:[int(x[0]),x[1],int(x[2])])\
    ,["id","name","wt"])
```

These 2 are attributes of the vertices

Marvel Social Network

GraphX

```
vertices=sqlC.createDataFrame\
    (sc.textFile(verticesPath)\
    .map(lambda x:x.split("|"))\
    .map(lambda x:[int(x[0]),x[1],int(x[2])])\
    ,["id","name","wt"])
```

The name of the character and the number of times he appears in the books

Marvel Social Network

GraphX

```
vertices=sqlC.createDataFrame\
    (sc.textFile(verticesPath)\
    .map(lambda x:x.split("|"))\
    .map(lambda x:[int(x[0]),x[1],int(x[2])])\
    ,["id","name","wt"])
```

Load the data from file into an RDD, parse the rows and give it to createDataFrame

Marvel Social Network

GraphX

Now we can create a GraphFrame

```
from graphframes import *
marvelSocial=GraphFrame(vertices, edges)
```

Marvel Social Network

GraphX

GraphFrame

has a bunch of interesting methods and attributes

Marvel Social Network

GraphX

GraphFrame

The degrees attribute will tell us the number of connections a vertex has

Marvel Social Network

GraphX

GraphFrame

Use the degrees attribute
to find the most important
(well-connected) characters

Marvel Social Network GraphFrame

GraphX

```
degrees=marvelSocial.degrees
```

```
degrees.take(10)
```

```
[Row(id=2031, degree=822),
 Row(id=3831, degree=38),
 Row(id=3231, degree=648),
 Row(id=3031, degree=352),
 Row(id=1831, degree=120),
 Row(id=2831, degree=188),
 Row(id=2631, degree=76),
 Row(id=3431, degree=46),
```

degrees is another DataFrame

Marvel Social Network GraphFrame

GraphX

```
degrees=marvelSocial.degrees
```

```
degrees.take(10)
```

```
[Row(id=2031, degree=822),
 Row(id=3831, degree=38),
 Row(id=3231, degree=648),
 Row(id=3031, degree=352),
 Row(id=1831, degree=120),
 Row(id=2831, degree=188),
 Row(id=2631, degree=76),
 Row(id=3431, degree=46),
```

The character id

Marvel Social Network GraphFrame

GraphX

```
degrees=marvelSocial.degrees
```

```
degrees.take(10)
```

```
[Row(id=2031, degree=822),
 Row(id=3831, degree=38),
 Row(id=3231, degree=648),
 Row(id=3031, degree=352),
 Row(id=1831, degree=120),
 Row(id=2831, degree=188),
 Row(id=2631, degree=76),
 Row(id=3431, degree=46),
```

The number of connections for that character

Marvel Social Network GraphFrame

GraphX

```
degrees=marvelSocial.degrees
```

```
degrees.take(10)
```

```
[Row(id=2031, degree=822),
 Row(id=3831, degree=38),
 Row(id=3231, degree=648),
 Row(id=3031, degree=352),
 Row(id=1831, degree=120),
 Row(id=2831, degree=188),
 Row(id=2631, degree=76),
 Row(id=3431, degree=46),
```

We can sort the DataFrame by the degree column to get the Top 10 characters

Marvel Social Network GraphFrame

GraphX

```
top10=degrees.sort(col("degree").desc()).take(10)
```

DataFrames have a sort method

Marvel Social Network GraphFrame

GraphX

```
top10=degrees.sort(col("degree").desc()).take(10)
```

You can specify a column name and sort them by that column

Marvel Social Network GraphFrame

GraphX

```
top10=degrees.sort(col("degree").desc()).take(10)
```

Here we sort in descending order of the degree column

Marvel Social Network GraphFrame

GraphX

```
top10=degrees.sort(col("degree").desc()).take(10)
```

We can print out the character names for these ids

Marvel Social Network GraphFrame

GraphX

```
for character in top10:
    charId=character.id
    print vertices.filter("id="+str(charId)).first().name
```

```
[Row(id=2031, degree=822),
```

Each member of the top10 list is a Row Object

Marvel Social Network GraphFrame

GraphX

```
for character in top10:
    charId=character.id
    print vertices.filter("id="+str(charId)).first().name
```

Extract the character id from the Row

Marvel Social Network GraphFrame

GraphX

```
for character in top10:
    charId=character.id
    print vertices.filter("id="+str(charId)).first().name
```

vertices is a DataFrame

Marvel Social Network GraphFrame

GraphX

```
for character in top10:
    charId=character.id
    print vertices.filter("id="+str(charId)).first().name
```

The filter transformation for DataFrames is slightly different

Marvel Social Network GraphFrame

GraphX

```
for character in top10:
    charId=character.id
    print vertices.filter("id="+str(charId)).first().name
```

This transformation uses a string as a condition instead of a lambda function

Marvel Social Network GraphFrame

GraphX

```
for character in top10:
    charId=character.id
    print vertices.filter("id="+str(charId)).first().name
```

The string uses columns in the DataFrame directly

Marvel Social Network GraphFrame

GraphX

```
for character in top10:
    charId=character.id
    print vertices.filter("id="+str(charId)).first().name
```

This condition will filter the vertex corresponding to the character id

Marvel Social Network GraphFrame

GraphX

```
for character in top10:
    charId=character.id
    print vertices.filter("id="+str(charId)).first().name
```

Extract the first row which matches and print the name

Marvel Social Network GraphFrame

GraphX

```
for character in top10:
    charId=character.id
    print vertices.filter("id="+str(charId)).first().name
```

```
CAPTAIN AMERICA
SPIDER-MAN/PETER PARKER
IRON MAN/TONY STARK
THING/BENJAMIN J. GRIMM
WOLVERINE/LOGAN
MR. FANTASTIC/REED RICHARDS
HUMAN TORCH/JOHNNY STORM
SCARLET WITCH/WANDA MAXIMOFF
THOR/DR. DONALD BLAKE/SIGURD JARLSON II/JAKE OLSON/LOREN OLSON
BEAST/HENRY &HANK& P. MCCOY
```

These are the top 10 characters in the Marvel Social Network

Marvel Social Network

GraphX

The module has several interesting methods for implementing **Graph Algorithms**

Graph Algorithms

GraphX

- Breadth first Search
- PageRank
- Connected Components
- Triangle search

Graph Algorithms

GraphX

Breadth first Search

PageRank

Connected Components

Triangle search

These are a few of the standard Graph processing algorithms available as built-in methods