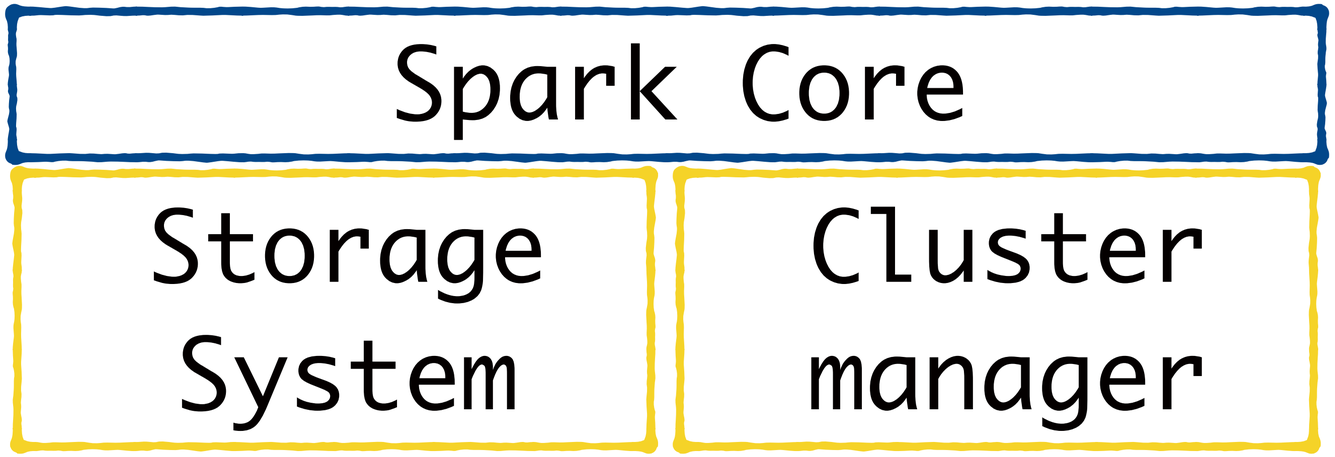
GRAPHX

Recap

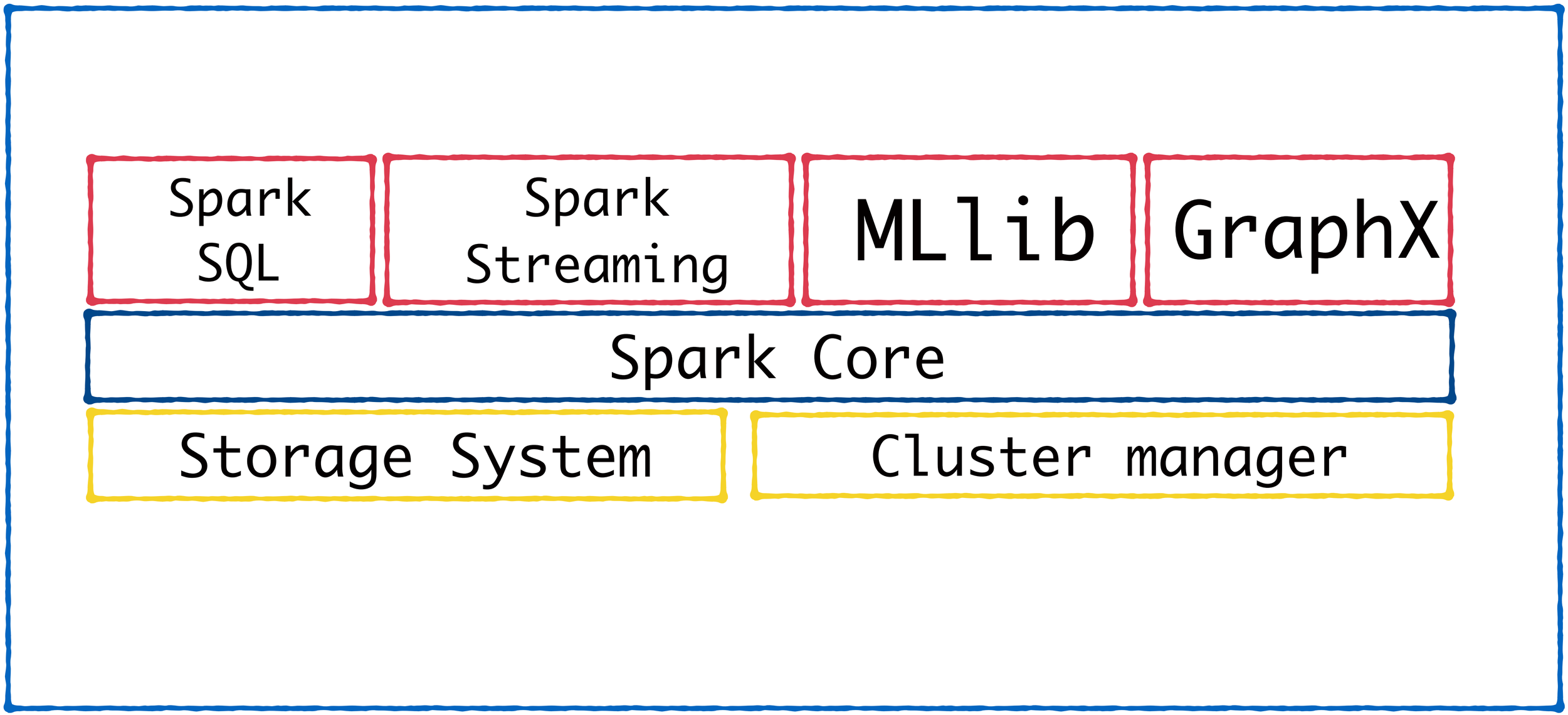
APACHE SPARK

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

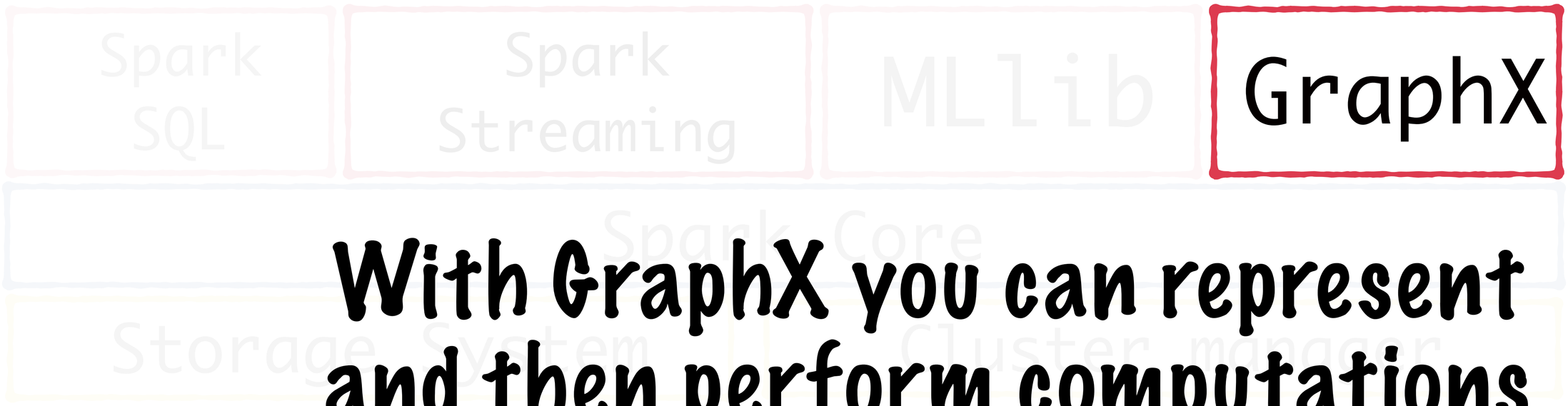
Spark Core

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

Let's use GraphX 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

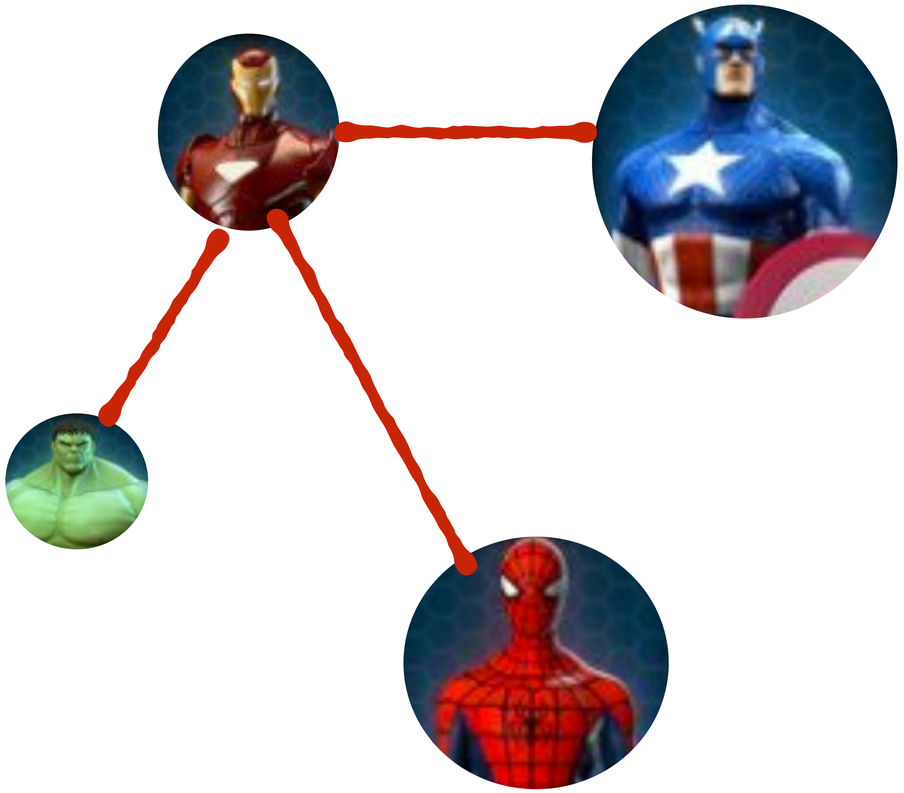
Marvel Social Network

GraphX

The graph would have

1. Characters as vertices

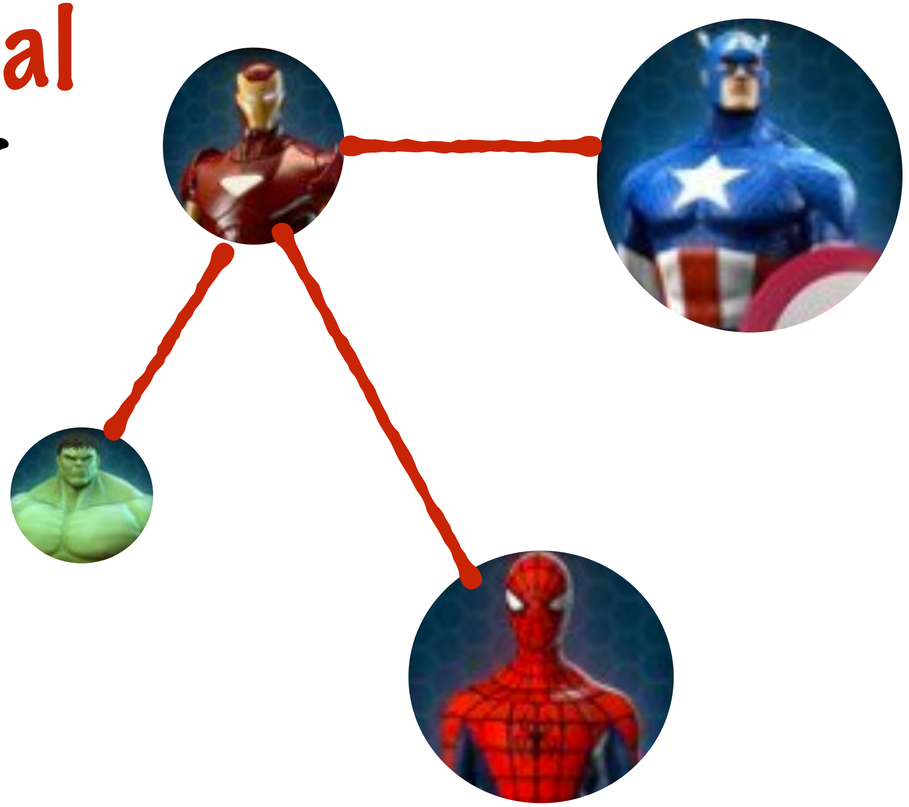
2. An edge representing if 2 characters appear together in a book



Marvel Social Network

GraphX

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

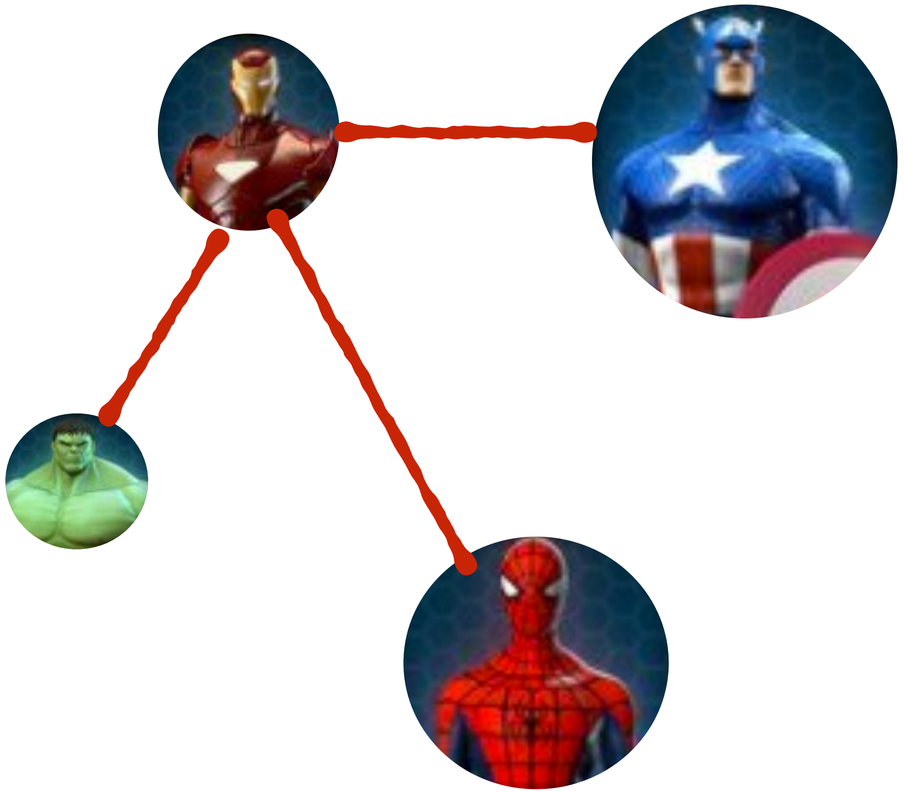


2. Edge weight is proportional to the number of times the characters appear together

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 RDD

An Edges RDD

Marvel Social Network

GraphX

A Vertices RDD

An Edges RDD

This RDD is made of tuples

(Vertex Id, Vertex Properties)

Marvel Social Network

GraphX

A Vertices RDD

(Vertex Id, Vertex Properties)

An Edges RDD

The Vertex Id must be a Long or a Hash Id

Marvel Social Network

GraphX

A Vertices RDD

(Vertex Id, Vertex Properties)

Vertex properties can have the Vertex name or a tuple with vertex properties . These are left to the user's discretion

Marvel Social Network

GraphX

A Vertices RDD "id" field

An Edges RDD

Each record must be an Edge object with
(Source,Destination,Edge Properties)

Marvel Social Network

GraphX

A Vertices RDD "id" field

An Edges RDD

(Source,Destination,Edge Properties)

Source and Destination must be Longs/
Hash ids corresponding to the Vertices

Marvel Social Network

GraphX

```
val gEdges = edges.
map(_.split(",")).
map(x => Edge(x(0).toLong,x(1).toLong,x(2).toInt))
```

Here is the edges DataFrame

Marvel Social Network

GraphX

```
val gEdges = edges.
map(_.split(",")).
map(x => Edge(x(0).toLong,x(1).toLong,x(2).toInt))
```

```
val edges = sc.textFile(edgesPath)
```

This is an RDD of Strings with what we need to represent the Edges

Marvel Social Network

GraphX

```
val gEdges = edges.
map(_.split(",")).
map(x => Edge(x(0).toLong,x(1).toLong,x(2).toInt))
```

Parse the rows into Arrays

Marvel Social Network

GraphX

```
val gEdges = edges.
map( _.split(",")).
map(x => Edge(x(0).toLong,x(1).toLong,x(2).toInt))
```

(Source,Destination,Edge Weight)

Set up Edge objects from the Arrays

Marvel Social Network

GraphX

```
import org.apache.spark.graphx._

val gEdges = edges.
map(_.split(" ")).
map(x => Edge(x(0).toLong,x(1).toLong,x(2).toInt))
```

Edge objects are part of the GraphX library

Marvel Social Network

GraphX

```
val gVertices = vertices.
map(_.split("[|]")).
map(x => (x(0).toLong,x.slice(1,x.length))).
distinct()
```

Similarly, we can set up a vertices RDD

Marvel Social Network

GraphX

```
val gVertices = vertices.
map(_.split("[|]")).
map(x => (x(0).toLong,x.slice(1,x.length))).
distinct()
```

This is an RDD of Strings with what we need to represent the Vertices

```
val vertices = sc.textFile(verticesPath)
```

Marvel Social Network

GraphX

```
val gVertices = vertices.
map(_.split("[|]")).
map(x => (x(0).toLong,x.slice(1,x.length))).
distinct()
```

Split the row into Arrays
(the string is pipe-separated)

Marvel Social Network

GraphX

```
val gVertices = vertices.
map(_.split("[|]")).
map(x => (x(0).toLong,x.slice(1,x.length))).
distinct()
```

(Vertex Id, Vertex Properties)

Set up a tuple to represent each Vertex

Marvel Social Network

GraphX

Now we can create a Graph

```
val socialGraph = Graph(gVertices,gEdges)
```

Marvel Social Network

GraphX

GraphX

has a bunch of interesting methods for Graphs

Marvel Social Network

GraphX

The degrees method will tell us the number of connections each vertex has

Marvel Social Network

GraphX

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

Marvel Social Network

GraphX

```
val socialDegrees = socialGraph.degrees
```

socialDegrees is another RDD with

(Vertex Id, Degrees)

Marvel Social Network

GraphX

```
val socialDegrees = socialGraph.degrees
```

Vertex Id represents a character

Marvel Social Network

GraphX

```
socialDegrees.
join(gVertices).
map(x => (x._2._2(0).trim,x._2._1)).
distinct().
sortBy(-_._2).
take(10)
```

Print the top 10 characters based on the degree

Marvel Social Network

GraphX

```
socialDegrees.
join(gVertices).
map(x => (x._2._2(0).trim,x._2._1)).
distinct().
sortBy(-_._2).
take(10)
```

Join (VertexId, Degree)
with (VertexId, Properties)

Marvel Social Network

GraphX

```
socialDegrees.
join(gVertices).
map(x => (x._2._2(0).trim,x._2._1)).
distinct().
sortBy(-_._2).
take(10)
```

(VertexId, (Degree, Properties))

Extract the character name and degree

Marvel Social Network

GraphX

```
socialDegrees.
join(gVertices).
map(x => (x._2._2(0).trim,x._2._1)).
distinct().
sortBy(-_._2).
take(10)
```

(VertexId, (Degree, Properties))

Marvel Social Network

GraphX

```
socialDegrees.
join(gVertices).
map(x => (x._2._2(0).trim,x._2._1)).
distinct().
sortBy(-_._2).
take(10)
```

(VertexId, (Degree, Properties))

Properties is an Array in which the first element is the Character Name

Marvel Social Network

GraphX

```
socialDegrees.
join(gVertices).
map(x => (x._2._2(0).trim,x._2._1)).
distinct().
sortBy(-_._2).
take(10)
```

(VertexId, (Degree, (Name,Weight)))

Properties is an Array in which the first element is the Character Name

Marvel Social Network

GraphX

```
socialDegrees.
join(gVertices).
map(x => (x._2._2(0).trim,x._2._1)).
distinct().
sortBy(-_._2).
take(10)
```

(Name, Degree)

Remove duplicates if any

Marvel Social Network

GraphX

```
socialDegrees.
join(gVertices).
map(x => (x._2._2(0).trim,x._2._1)).
distinct().
sortBy(-_._2).
take(10)
```

(Name, Degree)

Sort in descending order of degree

Marvel Social Network

GraphX

```
socialDegrees.
join(gVertices).
map(x => (x._2._2(0).trim,x._2._1)).
distinct().
sortBy(- . 2).
take(10)
```

(Name, Degree)

Pick the top 10

Marvel Social Network

GraphX

```
socialDegrees.
join(gVertices).
map(x => (x._2._2(0).trim,x._2._1)).
distinct().
sortBy(-_._2).
take(10)
```

(Name, Degree)

```
Array((CAPTAIN AMERICA,3866), (SPIDER-MAN/PETER PARKER,3482), (IRON MAN/TONY STARK,3056), (THING/BENJAMIN J. GRIMM,2
52), (WOLVERINE/LOGAN,2788), (MR. FANTASTIC/REED RICHARDS,2772), (HUMAN TORCH/JOHNNY STORM,2742), (SCARLET WITCH/WAN
A MAXIMOFF,2690), (THOR/DR. DONALD BLAKE/SIGURD JARLSON II/JAKE OLSON/LOREN OLSON,2578), (BEAST/HENRY &HANK& P. MCCO
Y,2560))
```

These are the top 10 characters in the Marvel Social Network

Marvel Social Network

GraphX

The GraphX 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