

UMD DATA605 - Big Data Systems

Graph Data Management Neo4J

Instructor: Dr. GP Saggese - gsaggese@umd.edu

TAs: Krishna Pratardan Taduri, kptaduri@umd.edu Prahar
Kaushikbhai Modi, pmodi08@umd.edu

v1.1

Overview

- Motivation
- Graph data models
- Storing graph data
- Querying graph data
- Typical graph analysis tasks
- Executing graph analysis tasks

Overview

- Motivation
- Graph data models
 - E.g., RDF, Property Graph, XML
- Storing graph data
 - E.g., Neo4j
- Querying graph data
 - E.g., Cypher, SPARQL, Gremlin
- Typical graph analysis tasks
 - E.g., PageRank, clustering
- Executing graph analysis tasks
 - E.g., Google Pregel, Apache Giraph, Spark GraphX

Overview

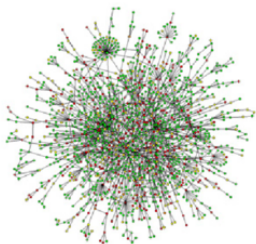
- **Motivation**
- Graph data models
- Storing graph data
- Querying graph data
- Typical graph analysis tasks
- Executing graph analysis tasks

Graphs: Background

- A *graph* (or* *network*)* captures a set of entities and interconnections between pairs of them
 - Entities / objects represented by *vertices* or *nodes*
 - Interconnections between pairs of vertices called *edges* (or *links*, *arcs*, *relationships*)
- Graph theory and algorithms widely studied in Computer Science
 - Not as much work on managing graph-structured data

Graph Data Structures: Motivation

- Increasing interest in querying and reasoning about the *underlying graph structure* in a variety of disciplines



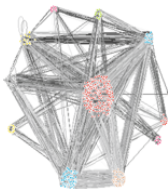
A protein-protein interaction network



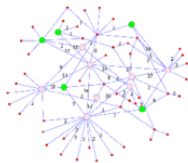
Social networks



Supreme court citation network

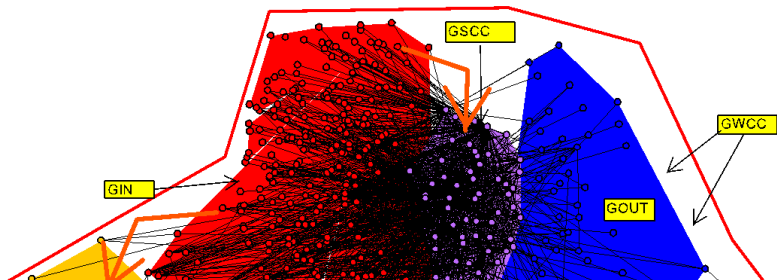
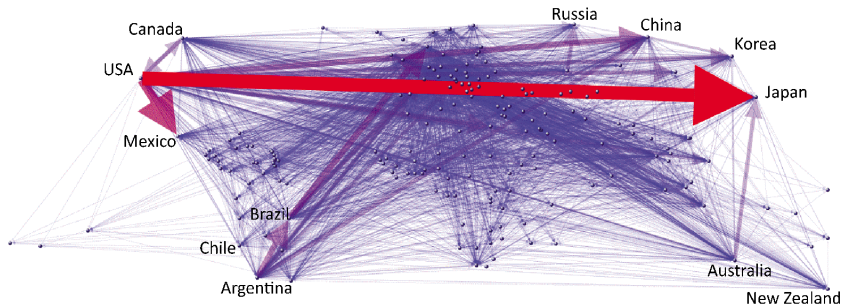


Financial transaction networks



Stock Trading Networks

Motivation



Motivation

- **Graph data structures have not changed that much over time**
 - Same problems in representing the data in 1960s than today
- **What has changed in recent years**
 - Large data volumes and easier availability
 - Reasoning about the graph structure can provide useful and actionable insights
 - Lose too much information if graph structure ignored
 - Not easy to query using traditional tools (e.g., relational DBs)
 - Need specialized tools (e.g., Neo4j)
 - Hard to efficiently process graph-structured queries using existing tools
 - Dedicated solutions: Google Pregel / Apache Giraph, Spark GraphX
 - Problems getting worse with increasingly large graphs seen in practice

Overview

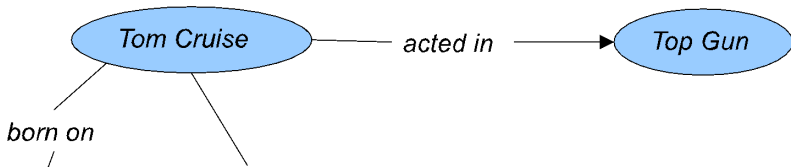
- Motivation
- **Graph data models**
- Storing graph data
- Querying graph data
- Typical graph analysis tasks
- Executing graph analysis tasks

Knowledge Graphs

- **Representation of knowledge in the form of graphs**
 - Capture entities, relationships, and properties
 - Provide a structured view of real-world information
- Can be represented using RDF or Property Graph models
 - E.g., Google Knowledge Graph, DBpedia, Wikidata
- **Applications**
 - Enable machine understanding of complex domains
 - Support semantic search, recommendation, and analytics
 - Used in various industries for data integration, knowledge discovery, and AI applications
- **Ontologies**
 - Provide a formal representation of knowledge
 - Promote interoperability across knowledge bases

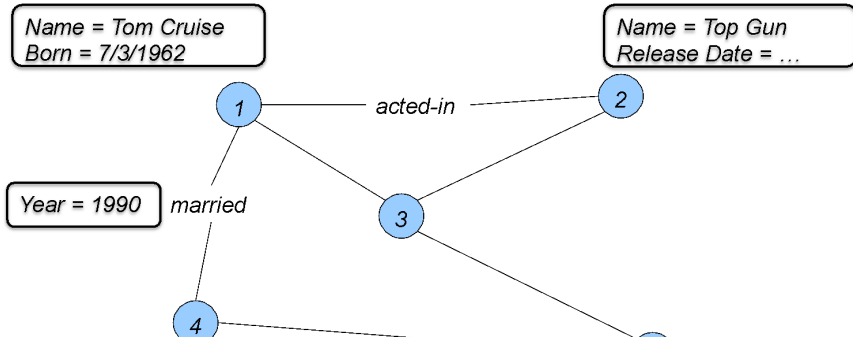
Graph Data Models: RDF

- Resource Description Framework
- RDF uses triples subject-predicate-object
 - Each triple connects a “subject” and an “object” through a “predicate”
 - E.g., “TomCruise-acted-TopGun”
- **Used to represent knowledge bases**
 - Typically queried through SPARQL
- **Pros**
 - Standardization
 - Standard W3C to model data
 - Subject and object can be URI (Uniform Resource Identifier) in semantic web
 - Interoperability
 - Can merge RDF data store
 - Extensibility
 - Can add new nodes and relationships
 - Support ontologies



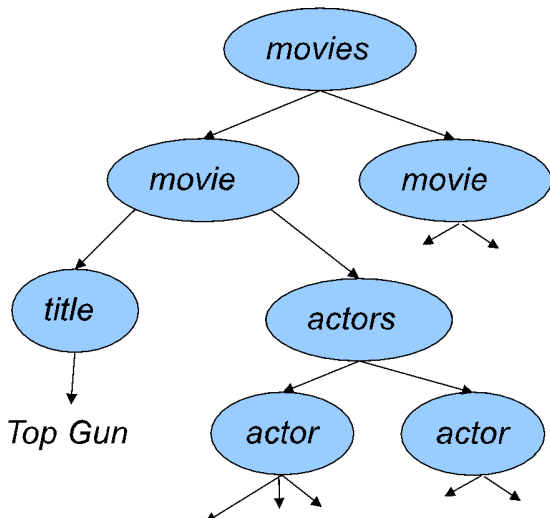
Graph Data Models: Property Graph

- A directed graph where each node and each edge may be associated with a set of *properties* (*key-values*)
- Query languages
 - Cypher (e.g., Neo4j)
 - Gremlin (e.g., Apache TinkerPop)
- Lack universal standard
- Similar expressive power to RDFs but less “schema” so more difficult to interoperate
- Used by many open-source graph data management tools



Graph Data Models: XML

- Commonly used data model for representing data without rigid structure
- It is a directed labeled tree
- Popular data exchange format for non-tabular data



Overview

- Motivation
- Graph data models
- **Storing graph data**
- Querying graph data
- Typical graph analysis tasks
- Executing graph analysis tasks

Storing Graph Data

- **File systems**
- Very simple
- No support for transactions, ACID
- Minimal functionality (e.g., must build the analysis/querying on top)
- **Relational database**
- Mature technology
- All the good stuff (SQL, transactions, ACID, toolchains)
- Minimal functionality
- **NoSQL key-value stores**
- Can handle very large datasets efficiently in a distributed fashion
- Minimal functionality
- **Graph database***
- Efficiently support for queries / tasks (e.g., graph traversals)
- Not as mature as RDBMs
- Often no declarative language (similar to SQL)
 - You need to write programs

Graph Databases

- Many specialized graph database systems in recent years
 - E.g., Neo4j, Titan, OrientDB, AllegroGraph
- A few key distinctions from relational databases
 - Built to manage and query graph-structured data
 - Store the graph structure explicitly using data structures with pointers
 - Avoid the need for joins, making graph traversals easier
 - More natural to write *queries* and *graph algorithms* (reachability or shortest paths)
 - Support graph query languages like SPARQL, Cypher, Gremlin
 - Fairly rudimentary declarative interfaces
 - Most applications need to be written using programmatic interfaces
 - Expose a programmatic API to write arbitrary graph algorithms

```text

## #### Overview

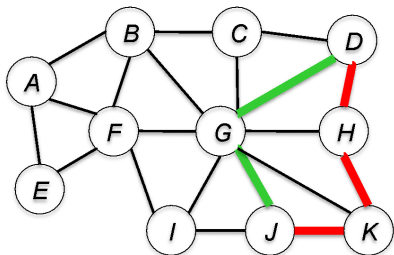
- Motivation
- Graph data models
- Storing graph data
- **Querying graph data**
- Typical graph analysis tasks





# Queries: Connection Subgraphs

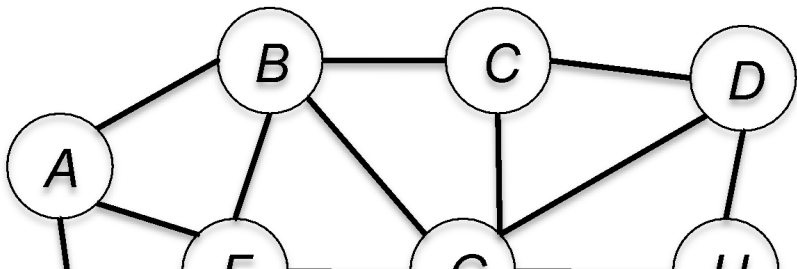
- Given a data graph and two (or more) nodes in it, find a small subgraph that best captures the relationship between the nodes
- How to define “best captures”?
  - E.g., “shortest path”: but that may not be most informative



*The “red” path between D and J  
maybe more informative than the  
“green” path*

# Graph Analysis: Centrality Measures

- Centrality measure: a measure of the relative importance of a vertex within a graph
- Many different definition of centrality measures
  - Can give fairly different results
- Degree centrality of a node
  - Number of edges incident on
- Betweenness centrality of a node
  - Number \*of shortest paths between pairs of vertices that go through
- Page Rank of a node
  - Probability that a random surfer (who is following links randomly) ends up at node



# Graph Analysis: Community Detection

---

- Goal: partitioning the vertices into (potentially overlapping) groups based on the interconnections between them
  - Basic intuition: More connections within a community than across communities
  - Provide insights into how networks function; identify functional modules; improve performance of Web services; etc.
- Numerous techniques proposed for community detection over the years
  - Graph partitioning-based methods
  - Maximizing some “goodness” function
  - Recursively removing high centrality edges
  - And so on . . .

# Overview

---

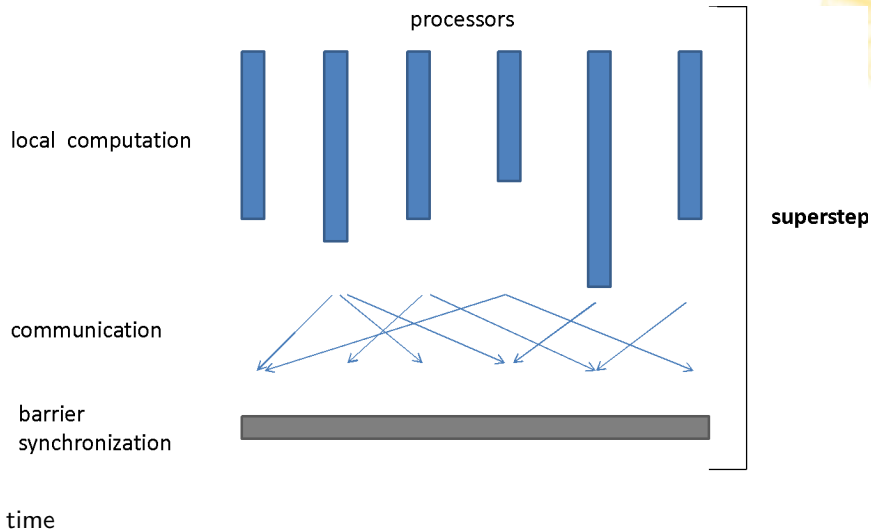
- Motivation
- Graph data models
- **Storing graph data**
- Querying graph data
- Typical graph analysis tasks
- **Executing graph analysis tasks**

# Bulk Synchronous Parallel (BSP)

---

- BSP model is a computational model used to design parallel algorithms for distributed systems
- Computation is divided into a series of *supersteps*, each consisting of three main phases
  - **Local computation phase**
    - Each processing unit performs calculations independently and concurrently, without any interaction
  - **Communication phase**
    - The processing units exchange information with each other by sending and receiving messages
    - These messages can be exchanged asynchronously without waiting for a response
  - **Synchronization phase**
    - Aka barrier
    - Ensures that all processing units have completed their local computations and communication before proceeding to the next superstep
    - This guarantees that all messages from the previous superstep have been received and processed
- Suitable for iterative graph algorithms
  - E.g., PageRank and Shortest Path

# Bulk Synchronous Parallel (BSP)



# Pregel System

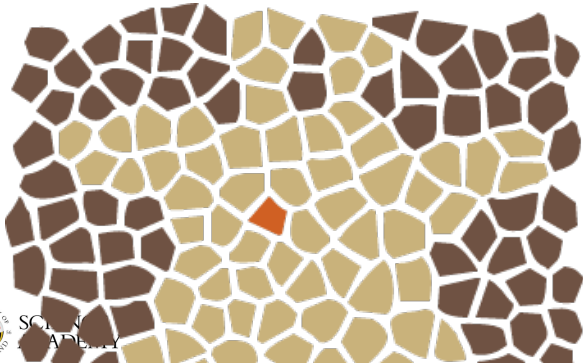
---

- Large-scale graph processing system developed by Google
  - Pregel paper, 2010
- Inspired by the Bulk Synchronous Parallel (BSP) model
  - Vertex-centric programming model
  - Asynchronous message passing between vertices
- Fault-tolerant using checkpointing mechanism
- Scalable and distributed architecture
- Designed for processing large graphs with billions of vertices and edges
- Handles graph mutations and updates during computation
- Not open-source, used internally at Google

# Apache Giraph

---

- Apache Giraph
- Open-source graph processing framework, inspired by Google's Pregel
- Implemented by Facebook and then open-sourced
- Built on top of Apache Hadoop
- Fault-tolerant using Hadoop checkpointing mechanism
- Scalable and distributed architecture
- Suitable for large-scale graph analytics and machine learning algorithms
- Actively maintained and widely adopted in the open-source community





# Apache Spark GraphX

---

- Apache Spark GraphX
- Graph processing library for Apache Spark
- Built on top of Spark's RDD (Resilient Distributed Dataset) model
- Supports both directed and undirected graphs
- Provides a flexible graph computation API
- Optimized for iterative graph computations
- Scalable and fault-tolerant architecture
- Supports in-memory graph processing for improved performance
- Suitable for large-scale graph analytics and machine learning tasks
- Implements various graph algorithms
  - E.g., PageRank, Connected Components, and Shortest Path

