LARGE SCALE NETWORK VISUALIZATION WITH GEPHI

ProjCINETViz

Authors:
Md Maksudul Alam
S M Arifuzzaman
Md Hasanuzzaman Bhuiyan

VIRGINIA TECH
CS 5604: Information Storage and Retrieval
Fall 2012

Table of Contents

Executive Summary 2
Project Overview 3
Gephi 3
  Why Gephi? 3
Network Representation 4
Difficulties and Implementation Challenges 4
Visualizing Large Networks 5
Future work 5
User Manual 5
Developer Manual 8
  CINET 8
  CINET System Architecture 8
  Blackboard 9
  Brokers 9
  CINET Interface 9
  Compute Resource 9
  CINET Existing User Interface – GRANITE 9
  Implementation of CINETViz in GRANITE 10
    Visualization Tab 10
  Development 12
    Java Classes 12
Workflow 14
Data Flow 15
Technologies Used 15
Reference 15

Executive Summary

The notion of graphs or networks is sufficiently pervasive since it can be used to model various types of data sources. Social, biological, and other networks capture the underlying structural and relational properties. Analysis of different networks reveals interesting information of the corresponding domain or system. Network analysts, thus, strive to analyze various networks by applying different algorithms and try to connect obtained insights to make sense of a unified theme, pattern or structure. For example, analysis of facebook friend network of a person can reveal information such as, groups of highly clustered people, most influential person in terms of connections, connecting persons between different cluster of people, etc.

While analyzing networks and digesting the information therein, analysts gradually form internal mental models of the people, places, events, or any sort of entity represented in the networks. As the number of nodes grows larger, however, it becomes increasingly difficult for an investigator to track the connections between data and make sense of it all. Many researchers believe that visual representations of data can help people better examine, analyze, and understand them. Norman [Norman94] has described how visual representations can help augment people's thinking and analysis processes.

The objective of the project is to develop visual representations of nodes, edges, and labels of a network in order to help analysts search, review, and understand the network better. We seek to create interactive visualizations that will highlight and identify significance of nodes, cluster formation, etc., in the networks where entities may be, for example, people, places, webpage, biological entity, dates and organizations. Basically, we want to build visual representations of the networks that help analysts making sense by applying different algorithms on them and observe the difference of nodes and edges in terms of color, and size.

A very important aspect of the project is the integration of the visualization module with CINET [CINET2012], a cyberinfrastructure for network science. CINET includes a set of graph algorithms and various types of networks. Analysis of networks are done by applying algorithms on those networks; results are obtained as text files containing information of different measures of nodes or edges. Complex workflow is intended while working with CINET where output of one analysis can be used as input to further analysis. Visualization comes as a great aid when analyst want to filter his interest on some particular nodes or a portion of the graph and conduct subsequent analysis on the smaller part. Though there are some existing visualization tools, e.g., Jigsaw [Jigsaw08], Sentinel Visualizer, NetLens, etc., they are more focused on information representation rather than on graph exploration or summarization capabilities. To the best of our knowledge, our project is the only one which supports network visualization as a part of complex workflow within network analysis utilizing high performance computing environment.

In summary, this project develops a visualization component for a VT digital library containing large network graphs (e.g., social networks and transportation networks). The visualization service will get datasets from an existing DL, visualize the graphs using Gephi (a java-based visualization library), and integrate the results within an NSF supported cyberinfrastructure (CINET).

Project Overview

We have developed a visualization module that can visualize graphs or networks using Gephi. This module is integrated with CINET and it supports large network graphs visualization. We have used Gephi toolkit library for the network visualization and gexf format of network representation. It supports visualization of different graph formats including directed, undirected, weighted, unweighted, labeled, unlabeled, etc.

Gephi

Gephi is an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs. It can visualize all kinds of networks including directed, undirected, weighted, un-weighted, labeled, un-labeled, etc. Gephi is compatible with various operating systems such as, Windows, Linux and Mac OS X. It is open-source and free as well. We will be using Gephi toolkit library to visualize networks in the web browser.

Why Gephi?

Similar types of other tools are available to visualize the networks (i.e., Cytoscape, Graphviz, Tulip). There are some certain advantages of using Gephi over other tools to visualize networks that we describe below.

- Input
  - Supports various types of networks (i.e., directed, undirected, mixed graphs, etc)
- Processing
  - Supports many network layout algorithms
  - Provides dynamic filtering option
  - Visualize networks in real time
- Output
  - Clustering and hierarchical representation of graph
  - Exports to multiple formats including PDF, PNG, SVG, etc.
- Extensibility
  - Modular and easily extensible through plugins
- Limitation
  - visualize network with 50K nodes and 1M edges.

Network Representation

We used the gexf format of network representation. It supports many graph features and this format is supported by numerous graph visualization tools. A comparison of the supported features by gexf is given below.

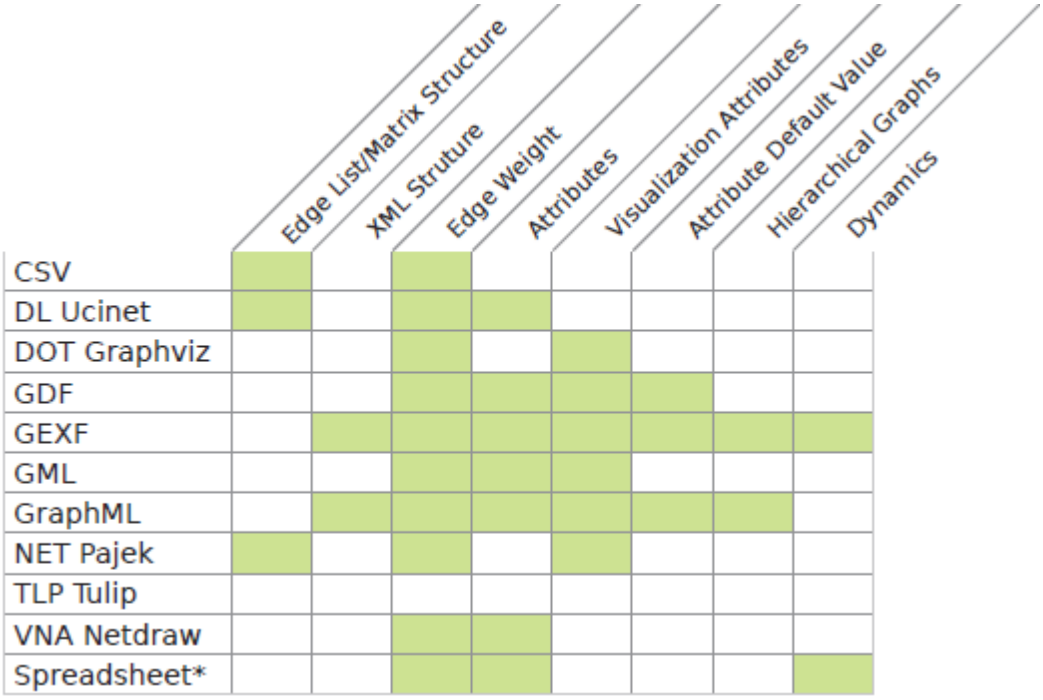


Fig 1: Comparison of gexf and other network representation fomrats **[GEXF]**

Difficulties and Implementation Challenges

The followings are the difficulties and challenges that we faced during implementation.

- Supporting different graph formats
  - Diverse
  - Conversion from one format to the standard format
- Data transfer from server to web app
  - Latency, bandwidth, browser compatibility and support
- Integration with CINET
  - Compatibility with existing architecture
  - Issues with smart-gwt etc.
- Study of CINET GRANITE framework
- Integration of visualization toolkit into web browser
  - Communicate between GWT and sigmajs visualization library using native javascript
- Communication between web server and high performance cluster

- Implementation of visualization methods (coloring, sizing, layouting) using gephi-toolkit programmatically

Visualizing Large Networks

We consider a network as large if the number of nodes |V| >= 10,000 or the number of edges|E| >= 50,000. As Gephi has limitations of visualizing large networks, we follow the following strategy to visualize only a small part in the case of such large networks.

- Choose a root node
  - Randomly
  - User defined
- Using BFS, explore from root up to:
  - Pre-specified depth (i.e., 4 or 5)
  - Pre-specified number of nodes (i.e., 200 nodes)
- Visualize the small portion of the network obtained by the above steps

Future work

As we will be continuing our work to develop a full version of visualization in CINET, the followings are the future work that we plan to finish.

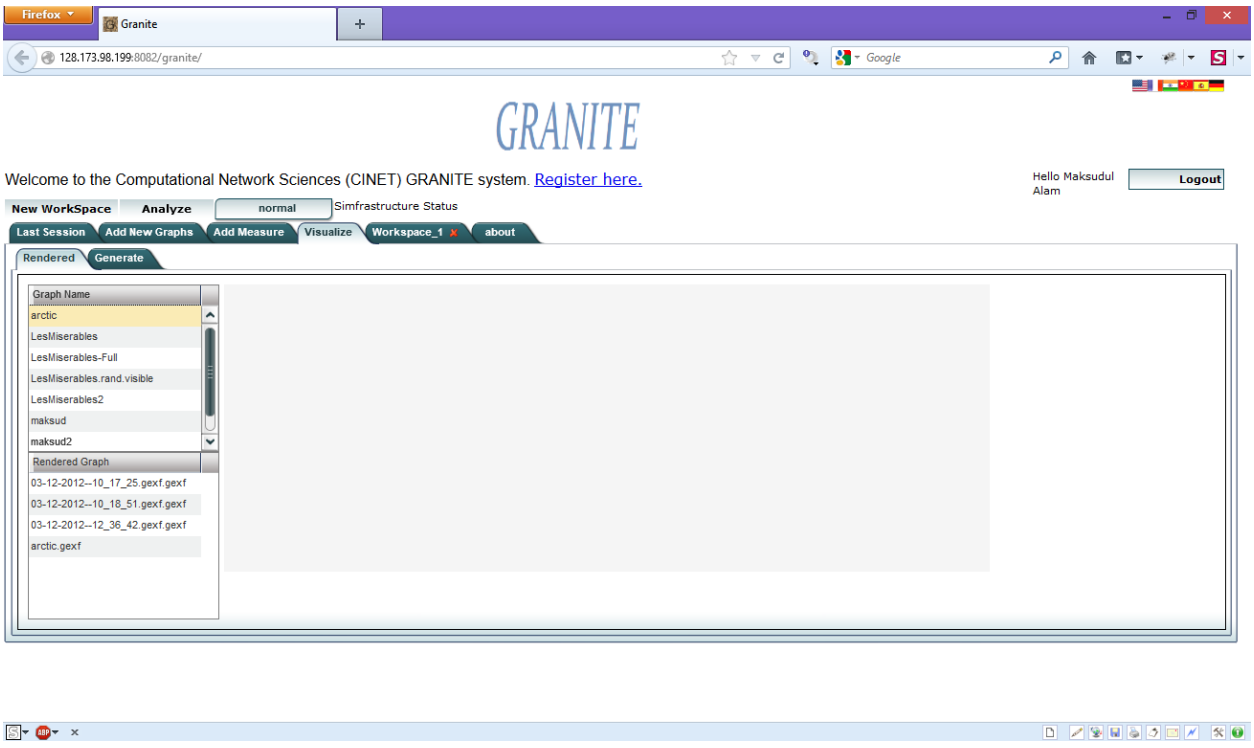
- Integrate it as a part of workflow
  - Visualizing the output
- Providing more information
  - Showing node label, id, edge weight and etc.
- Filtering
  - Visualize small part of graph
- Graph organization by applying multiple algorithms
  - For example, we want to apply both page rank and betweenness centrality
- Comparison of the different visualization
  - Using different measures

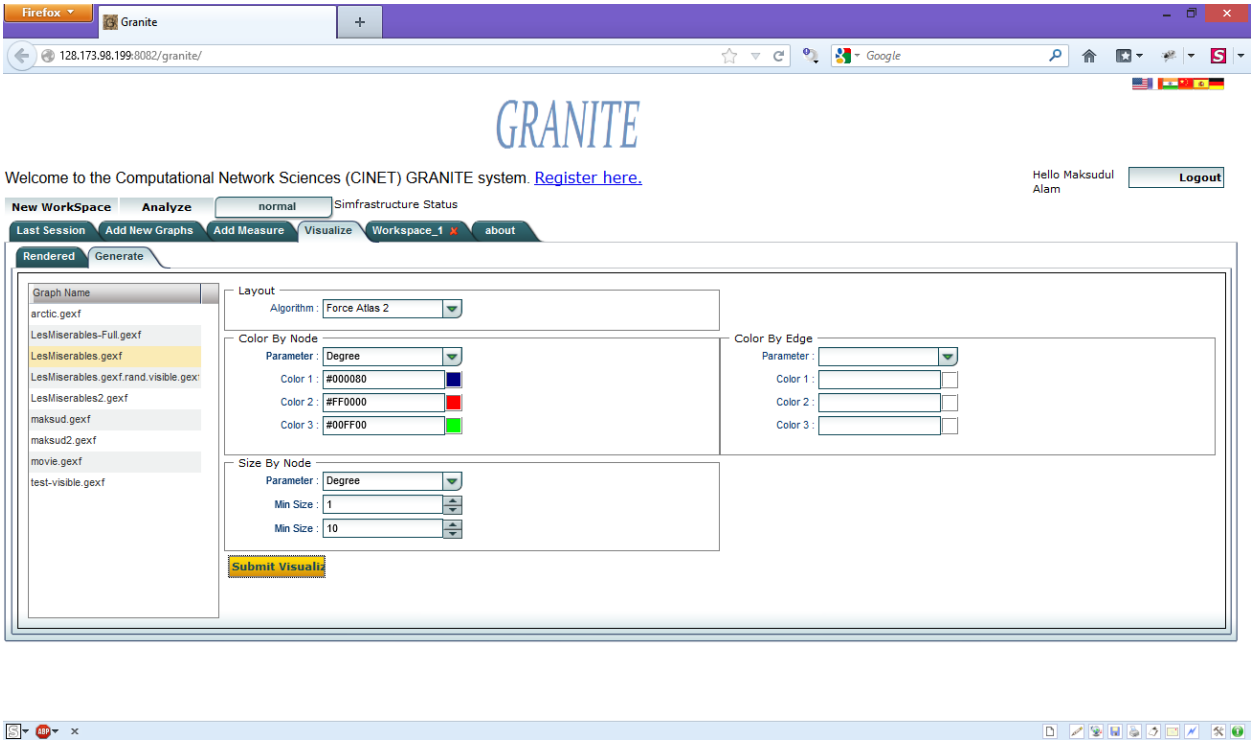
User Manual

Following steps should be followed to access and use the system:

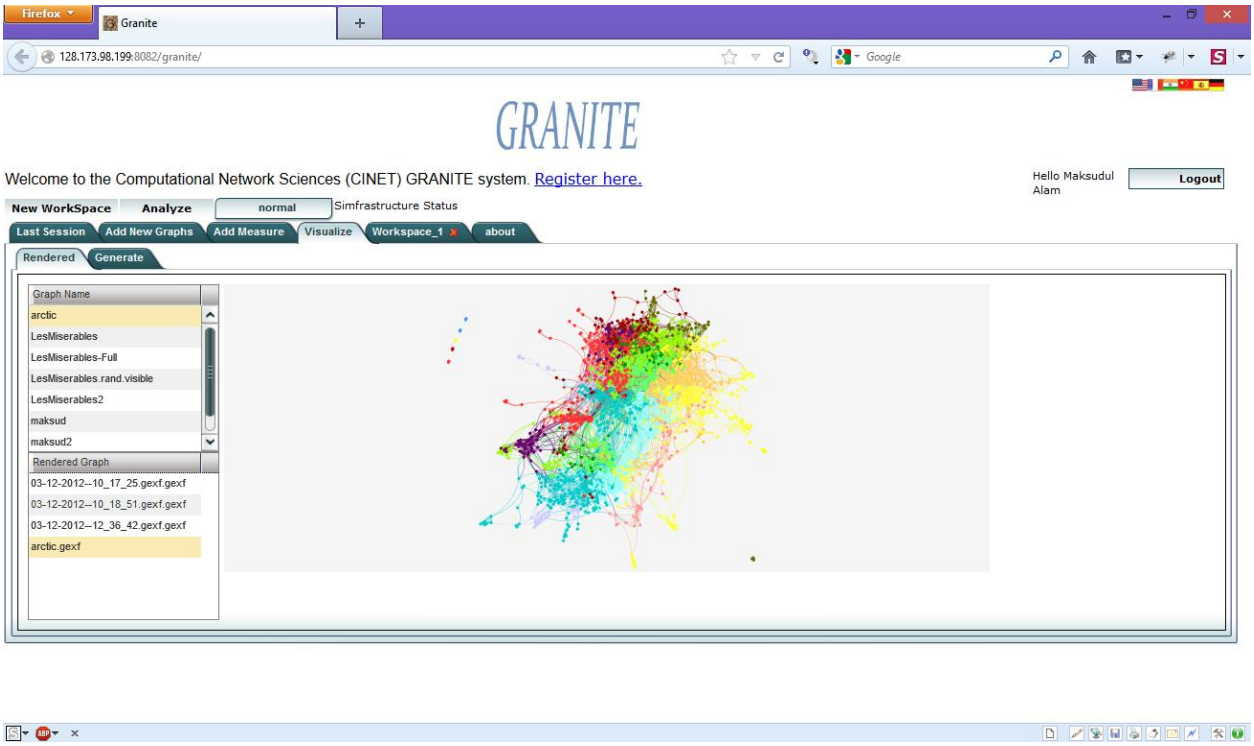
1. Registration: In order to be able to use the system, one needs to register for a GRANITE user account. The link for registration is as follow: ndssl.vbi.vt.edu/granite

2. Visualization Tab: Once logged in as a registered user, user needs to select the visualization tab. Now user can see a pre-rendered visualization or may want to generate visualization on the fly by selecting some parameters for visualization.





3. Export/Save: There is an option for saving rendered visualization in PDF or PNG format.



Developer Manual

CINET

CINET is a web portal that supports the complete set of user tasks related to cyber infrastructure (CI) content and resources. The system has two components: (i) the infrastructure component and (ii) the application component. The Granite system has the capability of generating graphs, computing graph measures. It has over 60 such network analysis operations. The Granite system contains 3 computation engine: (i) GaLib is developed in NDSSL lab, (ii) NetworkX [NetX], and (iii) SNAP [SNAP].

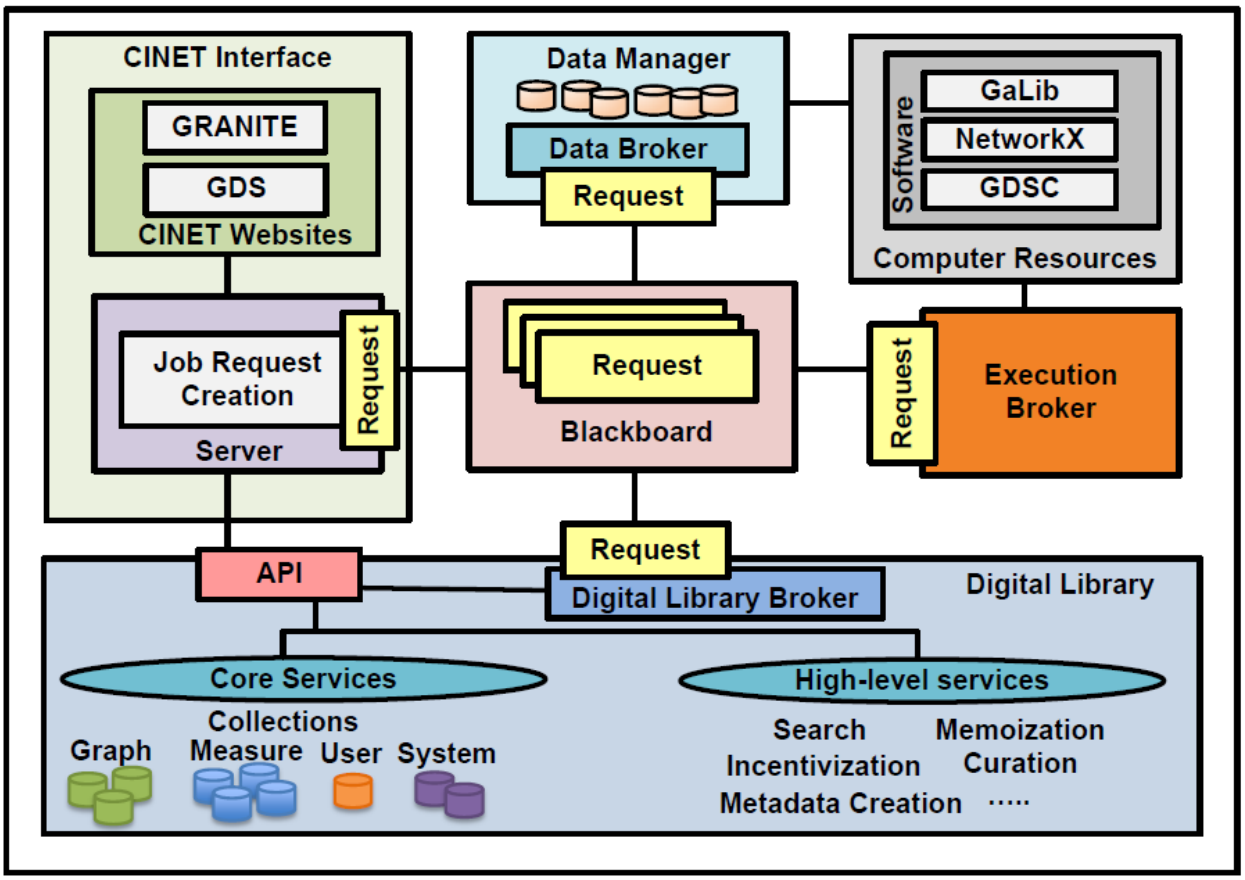


Fig 2: CINET System Architecture [CINET]

CINET System Architecture

CINET is a distributed system that supports a number of services that coordinate to fulfill a given request and perform its associated tasks. CINET embraces a JavaSpace based architecture and it uses persistent object exchange for coordination among services. Fig. 2 depicts the high-level architecture of the CINET framework.

Blackboard

The central communication and coordination system of CINET is the blackboard. It is currently implemented with a JavaSpace. It provides asynchronous, loose coupling of system components, being unaware of the existence of the other components in the system. They simply put requests onto the blackboard and wait for them to be fulfilled. Requests are Java objects that contain details about how it is to be fulfilled, in the form of an embedded workflow. For more details of the mechanism, see [CINET].

Brokers

A broker is responsible for providing a service in CINET. It monitors the blackboard for specific requests that it can fulfill and takes these requests from the space, executes the workflow embedded in the request.

CINET Interface

The CINET Interface consists of user interfaces and web applications that enables a user to submit analysis requests, add graph measure software, add graphs, and/or perform administrative tasks.

Compute Resource

Compute resources are the physical resources on which jobs are executed. Current resources are two HPC Linux clusters (Pecos and Shadowfax) at Virginia Tech. A typical compute resource runs NetworkX and GaLib components (which constitute the Granite compute engine). These components contain the binaries for performing graph analyses.

CINET Existing User Interface – GRANITE

The Granite system includes a CI web application that supports the interactive UI and allows the users to work with a wide range of graphs and measures. The Granite system interacts with the user through a tab-based interface as shown in Fig. 3. Smart GWT is used for Granite development.

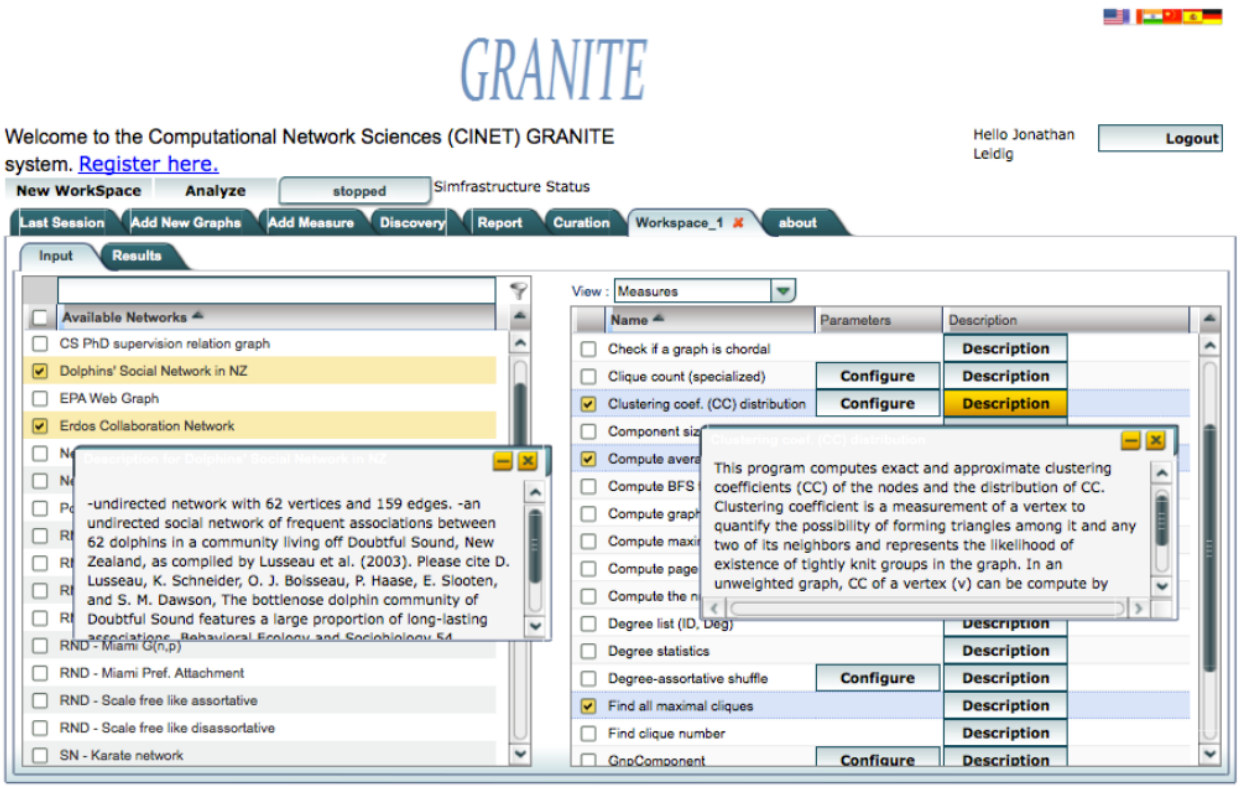


Fig 3: GRANITE user interface

Implementation of CINETViz in GRANITE

We have added a new tab in Granite named Visualization. This tab will contain our user interface to visualize networks. Figure 4 shows the appearance of the *Visualize* tab in the interface.

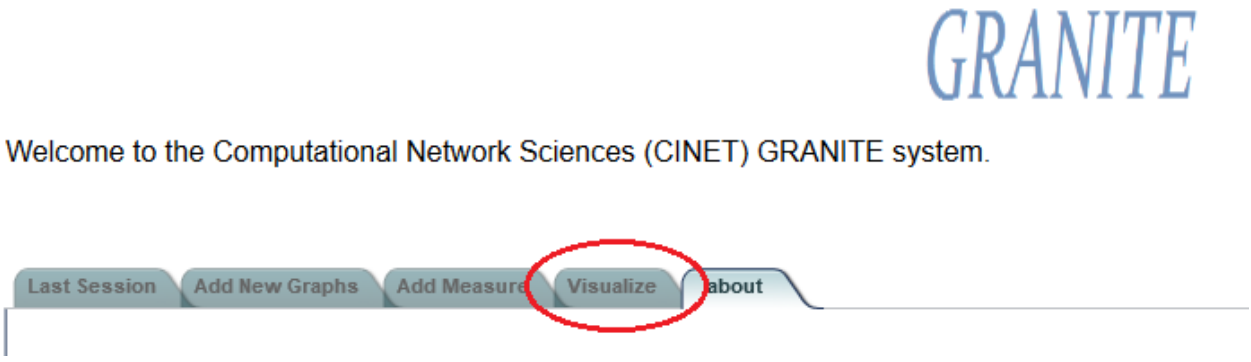


Fig 4: *Visualize* tab in the web interface

Visualization Tab

Visualization Tab contains two sub tab pages: Rendered and Generate. Rendered tab are to visualize the networks that is already generated and rendered networks. Generate tab is to generate networks for visualization from raw networks.

Rendered Tab

Figure 5 shows the rendered tab of CINETViz. Here we can actually visualize a network. There are three sections of importance here. In the first section marked as 1, shows the list of rendered networks. You can select any networks that is already rendered. We store all the rendered network in the web server for visualization.

Each network can have more than one visualization. These visualizations are shown in list marked 2. Whenever a user choose a network from first list the list of possible visualizations are shown in list 2.

When a user selects a visualization from list 2 he would see the visualization of the network in the section marked 3. This section is actually the main visualization toolkit. This toolkit is powered by Sigma javascript library. We communicate with the sigmajs library using Native Javascript API as defined by GWT. This toolkit has many customizable options which can be added as extensions. In our next iteration of the library we will implement the extensions.

We also allow the users to download the visualization of the network as a pdf. Clicking the download pdf version downloads the network as a pdf file. The pdf file is generated by gephi toolkit.

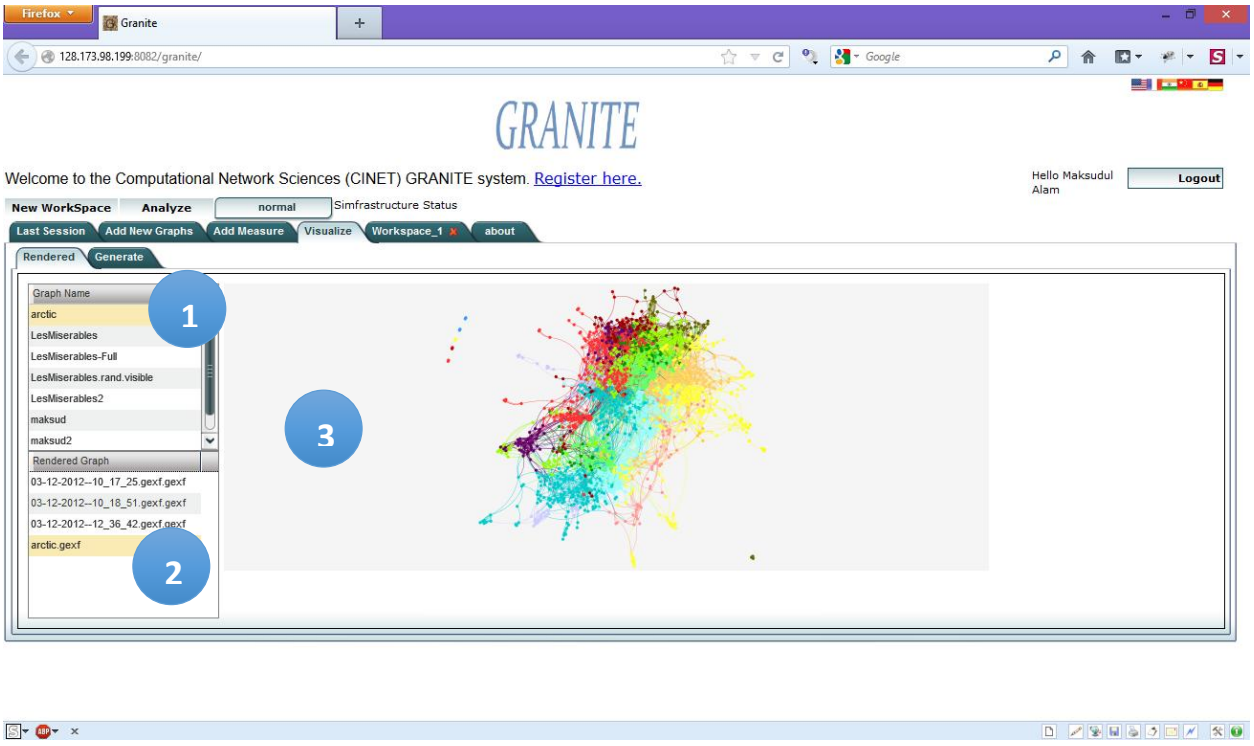


Figure 5: *Rendered* tab of the visualization panel in the interface

Generate Tab

In the generate tab we have the option of creating new visualizations from existing networks. Here user picks the networks first for visualizations and picks many visualization options. The visualization options provided in current version are given below:

A. Layouting: Users have the option to layout the network from a predefined set of options. These options includes: Random Layout, Force Atlas Layout and No Layout. The layout is provided by Gephi library and we have written an wrapper code to implement the layouting using gephi-toolkit.
B. Node Coloring: The user can choose node coloring based on various network node parameters. Currently we support Degree based parameters. User can also skip the node sizing, in this case existing node coloring scheme or default coloring scheme will be imposed.
C. Node Sizing: The node sizing can also be changed based on node parameters. We can run many network analysis tools on the network. For example we can run betweenness centrality and choose node sizing based on this. We support Degree and Betweenness Centrality measure in this version. We will continue to add more measures in next iteration of the system.
D. Edge Coloring: We also have the option of edge coloring from our system. We can change the color of the edges based on its weight. Also the edge line would be drawn based on the weight of the edges.

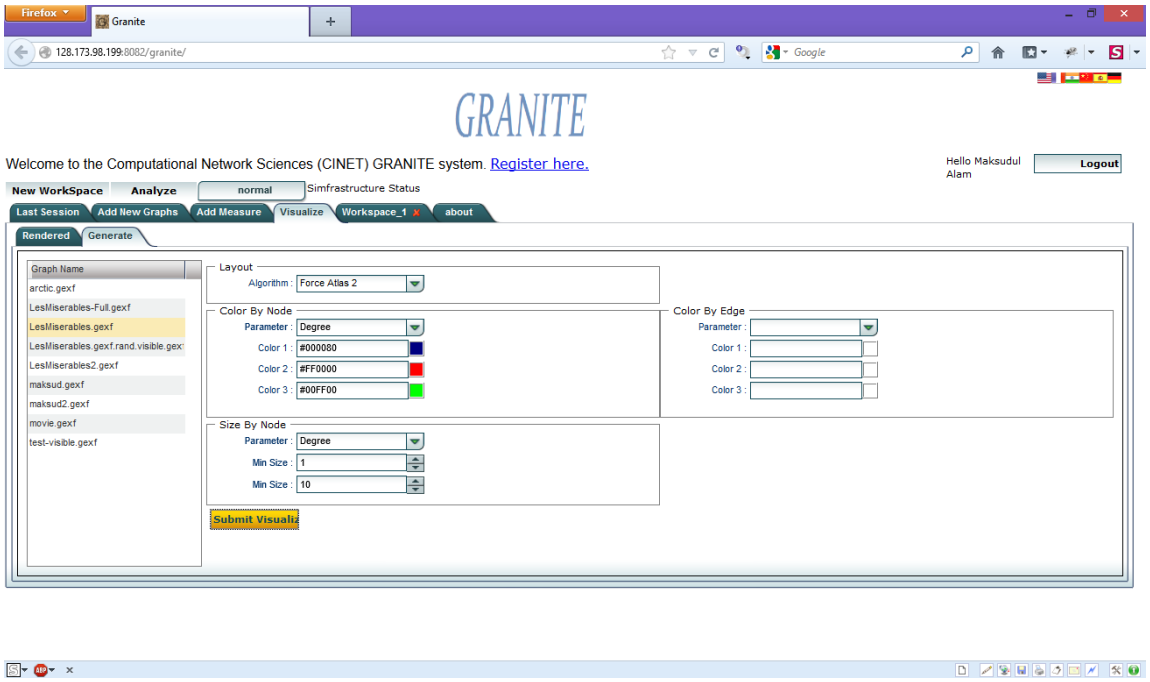


Figure 6: *Generate* tab of the visualization panel in the interface

Development

The development of CINETViz is based on existing GRANITE framework. We studied and extended the framework. In the framework our visualization tool is added as a tab. GRANITE is based on GWT. It has server and client component. Client component is converted into javascript code and Server component runs on the web server. We are using Jetty as the web server for our development purpose. In the following section we have listed our implemented classes.

Java Classes

The java classes developed for the implementation of CINETViz is shown in the following figures. Codes of the classes are provided separately in zipped folder.

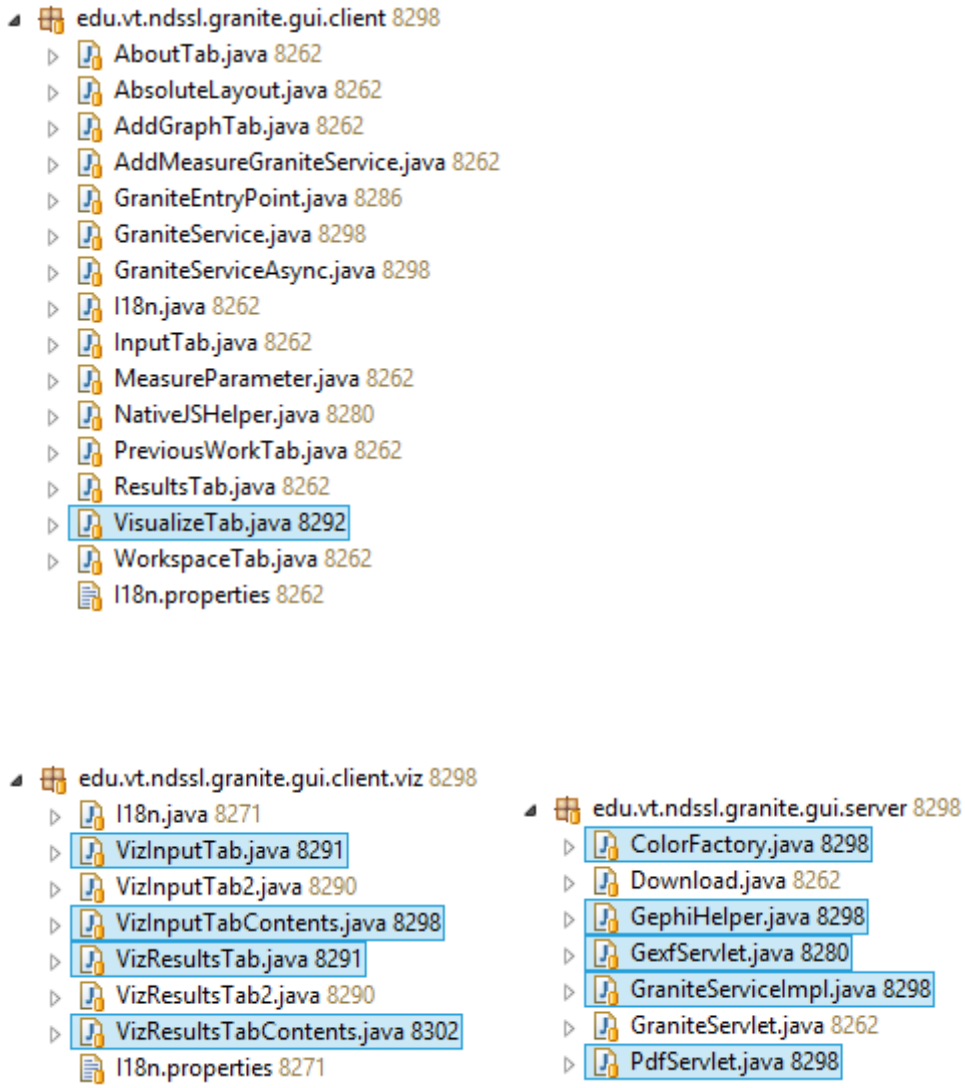


Fig 6: List of the java classes implemented

- **VisualizeTab.java**: VisualizeTab is extended from AbstractTab. It represents the Main visualization tab. The visualize tab has two sub tab pages: VizInputTab.java and VizResultsTab.java.
- **VizInputTab.java**: This file represents the Generate sub tab.
- **VizResultsTab.java**: This file represents the Rendered sub tab.
- **VizInputTabContents.java**: This is the main GUI component of Generate tab. It is a GWT composite.
- **VizResultsTabContents.java**: This is the main GUI component of Rendered tab. It is also a GWT composite tab.
- **GraniteService.java**, **GraniteServiceAsync.java**, **GraniteServiceImpl.java**: This three files are responsible for RPC communication between the servers and clients. All the webservices for CINETViz is implemented in these files.
- **GephiHelper.java**: this file is responsible for handling all gephi toolkit related interfacing. It contains the wrapper codes to call the gephi toolkit libraries. It handles the code for layouting, sizing, coloring the network.

- **GexfServlet.java**: This file is used by sigmajs library to get the rendered network file. It is basically a HttpSerlvet class.
- **PdfServlet.java**: This file is used to download the network as a pdf file. It is also a HttpServlet class.

Workflow

Visualization of a network involves the following steps:

1. User selects graph and parameters for visualization.
2. The graph is converted into gexf format which is sent to layout core for layouting.
3. After getting a layout, some network analysis algorithm is applied based on user selection to determine the color and/or size of the nodes and/or edges.
4. Finally the visualization is stored in the server and rendered on the web browser.

The CINETViz workflow is shown below. Here the used first generates a network by sending the networking parameters to the system. First the user goes to *generate* tab and selects the appropriate networking parameters. The parameters are converted into key-value Hash map. The hash map is sent to the server using GraniteService. Depending on the parameters the server applies layout, network analysis, color, size etc. After these are completed the network is stored in gexf format in the server according to our structure. Thus we store all rendered networks in the webserver.

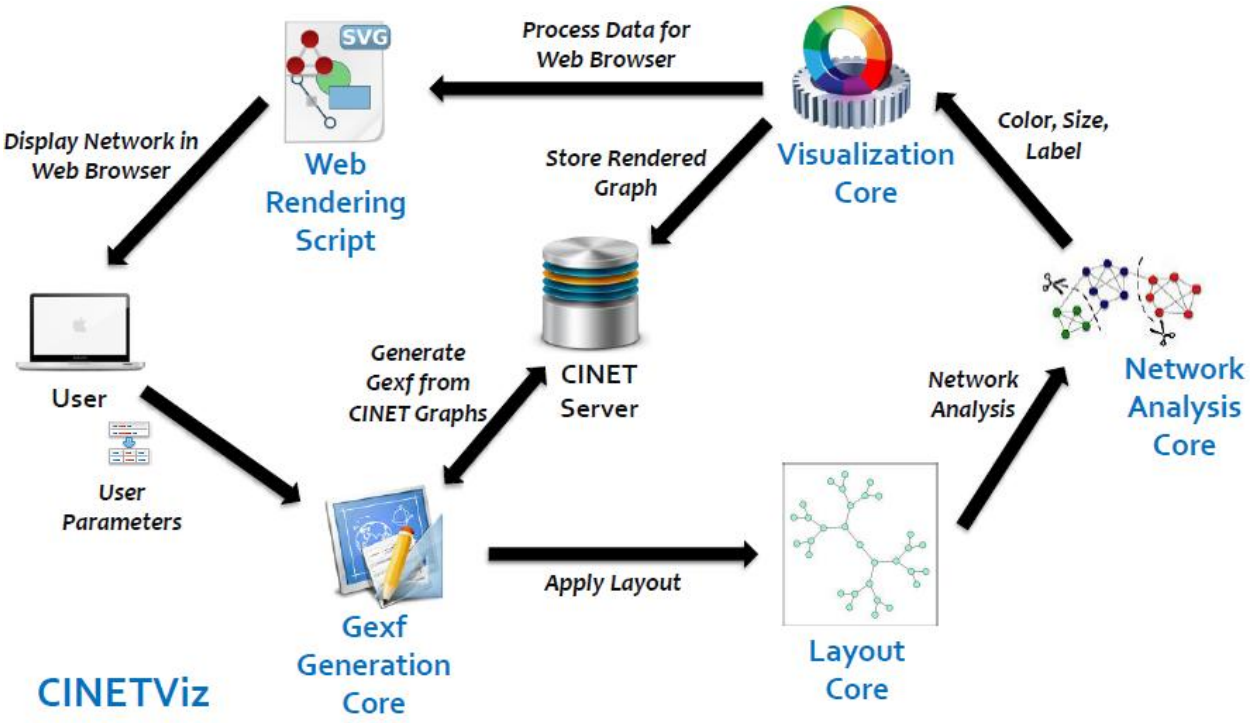


Fig 7: Typical visualization workflow

When the user requests a network visualization, a similar request is sent to the server and server returns with appropriate web path of the network visualization gexf docment using the GexfServlet. This is fed to the sigmajs library for final visualization.

Data Flow

Figure 8 shows the data flow of CINETViz. In CINETViz there is primarily two modules: Generate and Rendered Visualize. There are two main servers for GRANITE: The web server and the high performance cluster. The original network files are stored in the HPC cluster. To perform the operation we need to fetch the files from HPC cluster to GRANITE web server. We can do this in various ways, but the easiest way is using ftp. The Generate tab sends a request to GRANITE web server to fetch network files. The webserver in turns send a request to fetch files from HPC cluster. Once the files are fetched from HPC cluster to the web server the gephi operations are performed. The generated files are stored in the web server.

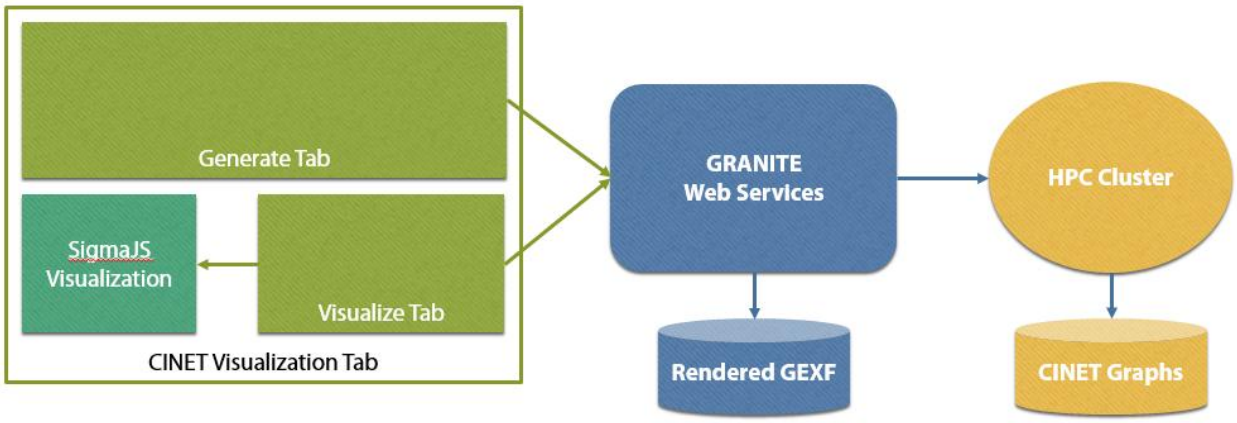


Fig 8: Data flow between visualization module and server

When the user request for previously rendered gexf files it is served directly from web server. Until this point we do not use database to store the information about rendered networks. It will be our priority task next iteration to include this support into GRANITE CINETViz.

Technologies Used

The following technologies and libraries are used for development of CINETViz:

- Gephi Toolkit
- SigmaJS visualization library
- Eclipse Juno
- Google Web Toolkit
- SmartGWT
- Jetty Web Server

Reference

[**Norman94**] Norman D. Visual representations. In Things That Make Us Smart: Defending Human attributes in the Age of the Machine. Addison-Wesley: Reading, MA, 1994.

[**CINET2012**] CINET: A CyberInfrastructure for Network Science, in eScience 2012, to be held on 8-12 Oct, 2012, Chicago.
[Jigsaw08] Jigsaw: supporting investigative analysis through interactive visualization, in Information Visualization (2008) 7, 118 -- 132
[GEXF] https://gephi.org/users/supported-graph-formats/
[GEPHI] https://gephi.org/
[NetX] http://networkx.lanl.gov/
[SNAP] htpt://snap.stanford.edu/