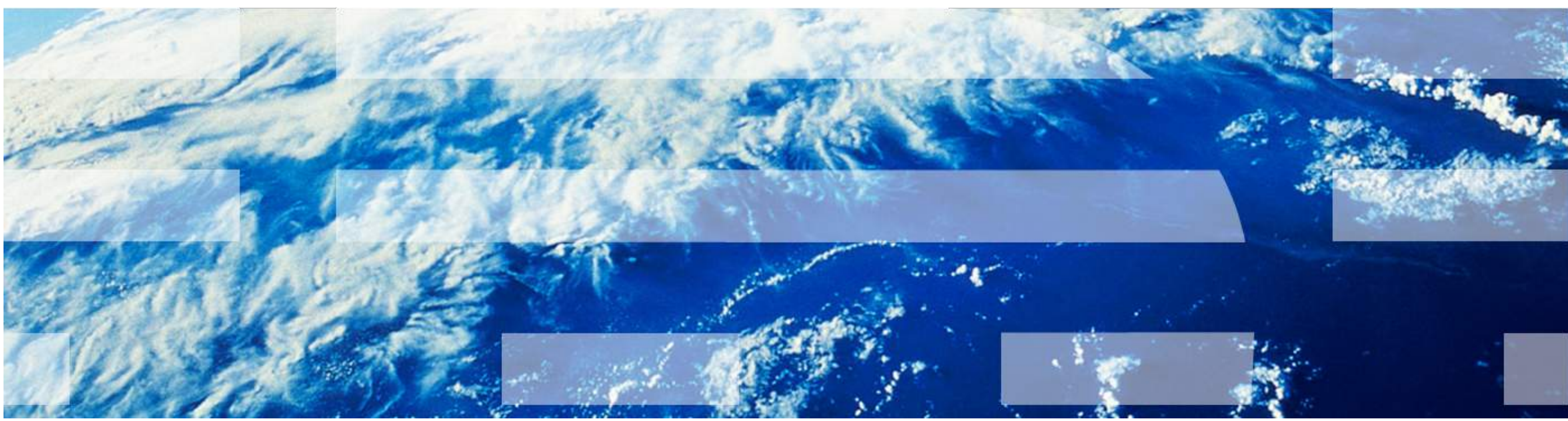
The World is Big and Linked: Whole Spectrum Industry Solutions towards Big Graphs

— Graph Computing and Tutorial of IBM System G

IBM System G Team

Presenters: Ching-Yung Lin (lead), Toyotaro Suzumura, Yinglong Xia

IBM T. J. Watson Research Center



October 31st, 2015

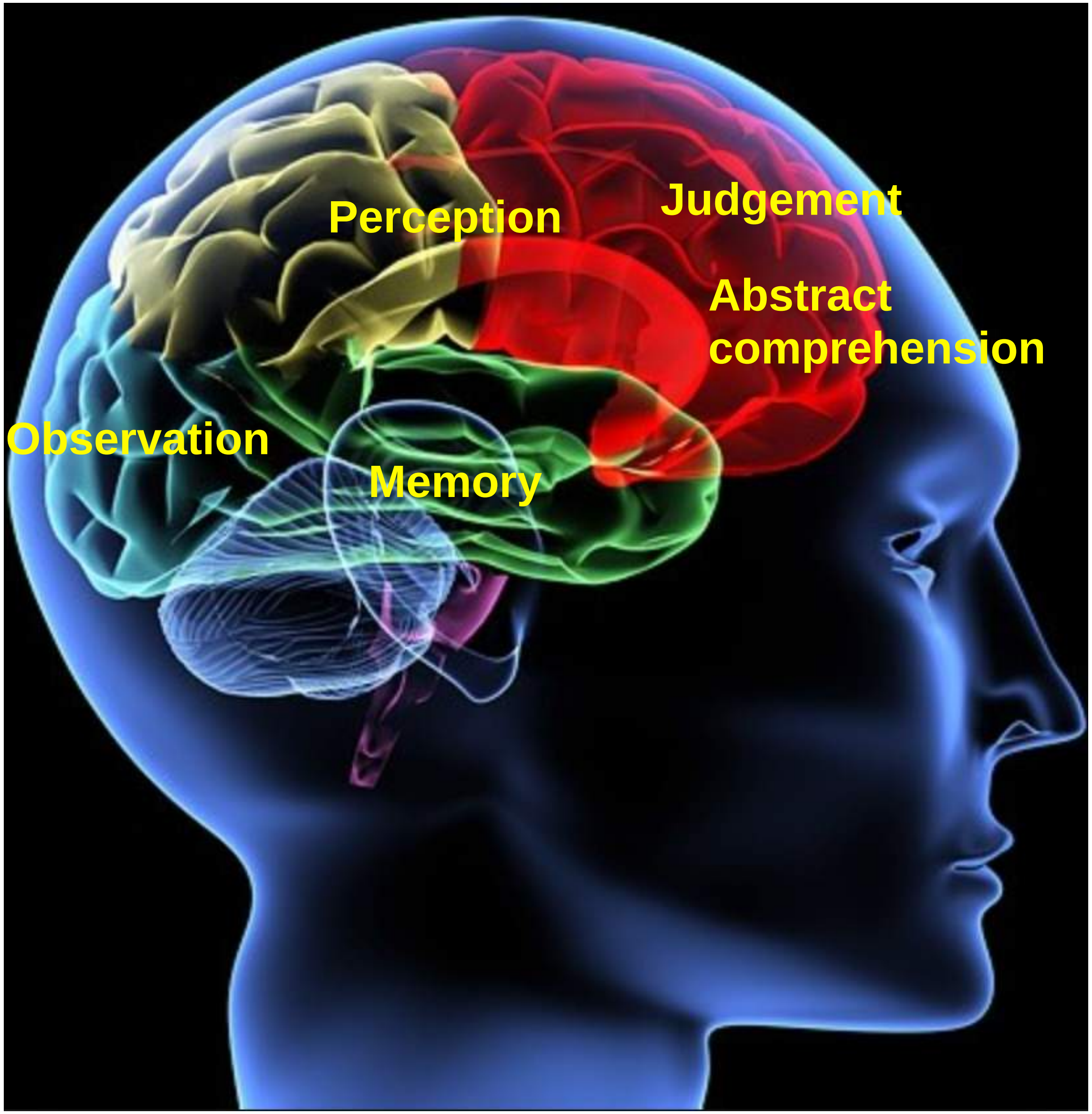
Agenda:

- 4:00 – 4:30 Introduction of IBM System G
- 4:30 – 4:40 IBM System G Visualizer & Demo
- 4:40 – 5:10 Quick Exploration of IBM System G
  - gShell, py-gShell, gremlin-gShell (groovy), REST API
  - gShell Analytics
  - Programming/User-Defined Plugins
- 5:10 – 5:50 Glance at IBM System G Eco-system
  - GraphBIG
  - ScaleGraph
- 5:50 – 6:00 Q&A

Introduction to IBM System G

http://systemg.research.ibm.com

Brain Functions and Evolution



Network / Graph is the way we remember, we associate, and we understand.

Demo

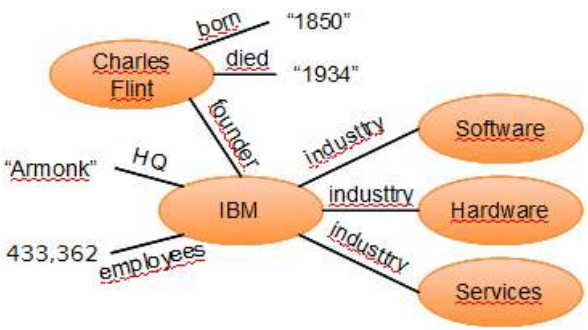
Joint project IBM System G team with Columbia Univ.

IBM System G — Graph Computing as an Intelligence Machine

Graph Database

Property Graph

Memory



subject	predicate	object
Charles Flint	born	"1850"
Charles Flint	died	"1934"
Charles Flint	founder	IBM
IBM	HQ	"Armonk"
IBM	employees	433,362
IBM	industry	Software
IBM	industry	Hardware
IBM	industry	Services

Related Information

Graph Analytics

Relation Graph

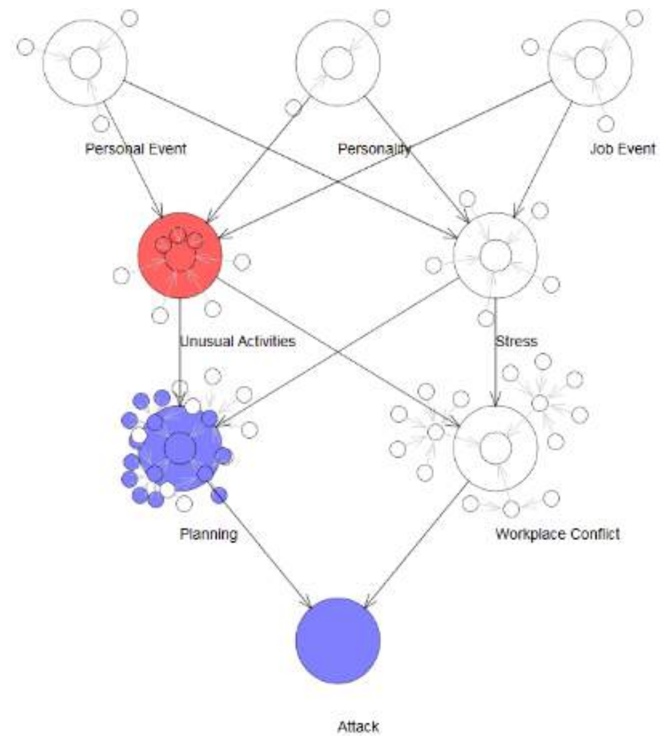
Perception

Contextual Analysis

Graphical Models

Reasoning Graph

Intelligence



Machine Reasoning & Deep Learning

http://systemg.research.ibm.com

System G Team

What is IBM System G?

IBM System G

IBM

A Complete Graph Computing Suite — Toolkits, Solutions, & Cloud

http://systemG.research.ibm.com (Internet) or http://systemG.ibm.com (IBM internal site)

Rich Graph Algorithm/ Functions Primitives

- Centralities
- Communities
- Graph Sampling
- Network Info Flow
- Shortest Paths
- Ego Net Features
- Graph Matching
- Graph Query
- Graph Search
- Bayesian Networks
- Latent Net Inference
- Markov Networks
- Spatio-Temporal Ana.

Multi Graph Type Support

- Few, very *large graphs* (e.g. social, Internet of things)
- Many, many *small graphs* (e.g. protein, healthcare)
- Large *semantic graph* (Semantic web, RDF, Graph search, Graph recommendation)
- Large *Probabilistic graphical models*: Bayesian networks, Markovian networks, HMMs, etc.

And More:

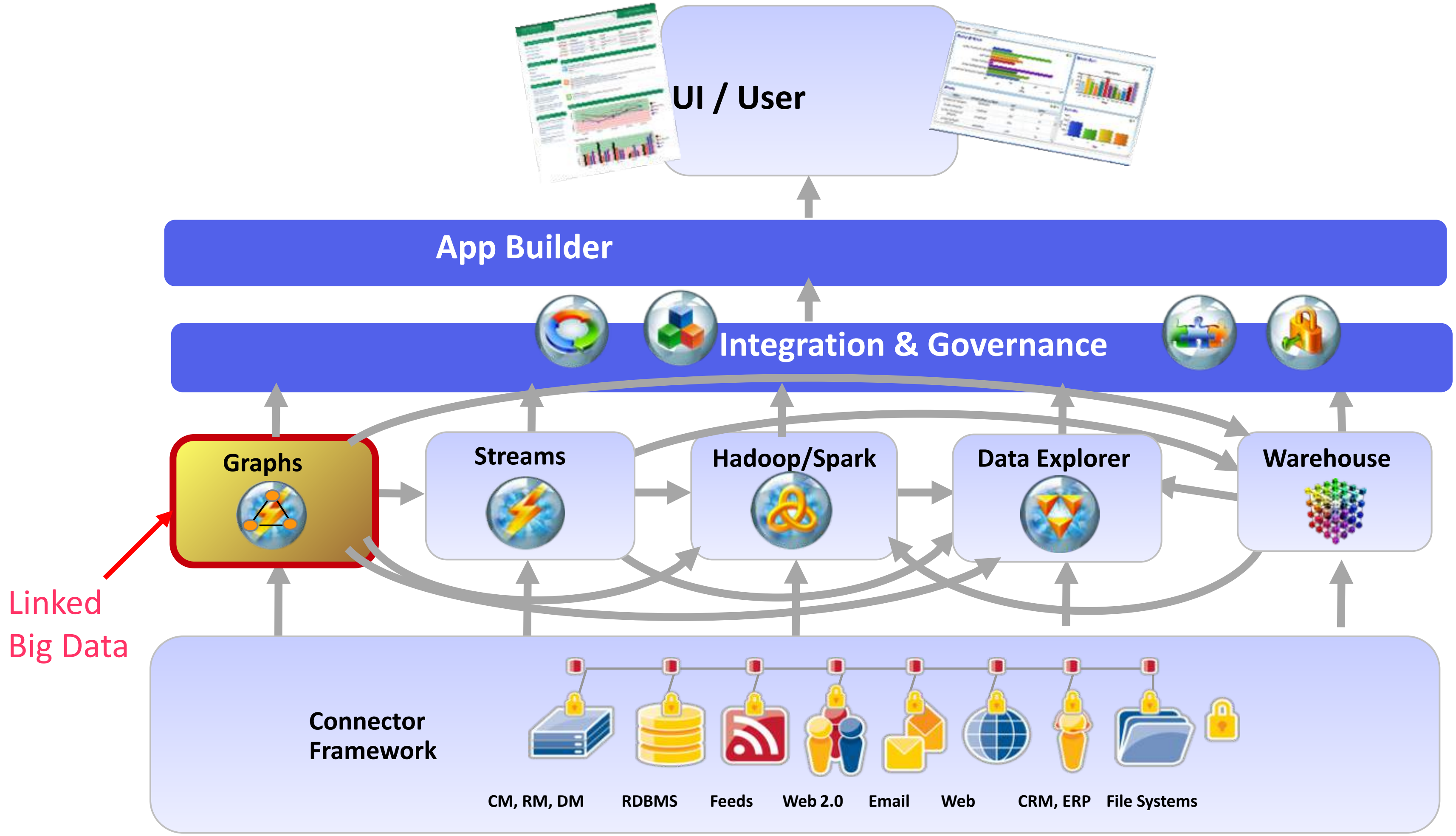
- Graph Visualizations
- Graph Databases
- Graph Middleware for Hardware Platform Optimization
- Cognitive Networks and Cognitive Analytics
- Graph-Enabled Industry Solutions

Based on 100+ innovations including 8 best paper awards; $22M+ R&D investment

"IBM System G" is an IBM corporate approved external naming (April 2014).
First production solution (SmallBlue): Oct 2008 ; Based on 30+ graph-related projects in IBM Research since 2003

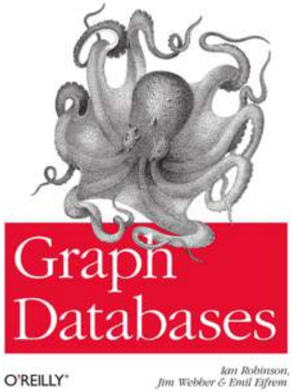
Big Data: "*While enterprises struggle to consolidate systems and collapse redundant databases to enable greater operational, analytical, and collaborative consistencies, changing economic conditions have made this job more difficult. E-commerce, in particular, has exploded data management challenges along three dimensions:* ***volumes*****,** ***velocity*** ***and*** ***variety****. In 2001/02, IT organizations much compile a variety of approaches to have at their disposal for dealing each*." – Doug Laney, Gartner, 2001

Graph is a missing pillar in the existing Big Data foundation



Graph Computing is difficult because data cannot be easily partitioned

Graph Database key differentiator — native store



In Relational DB, relationships are *distributed and stored as tables*

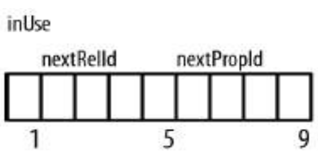
User					
UserID	**User**	**Address**	**Phone**	**Email**	**Alternate**
1	Alice	123 Foo St.	12345678	alice@example.org	alice@neo4j.org
2	Bob	456 Bar Ave.		bob@example.org	
...	...	...	...	...	...
99	Zach	99 South St.		zach@example.org	

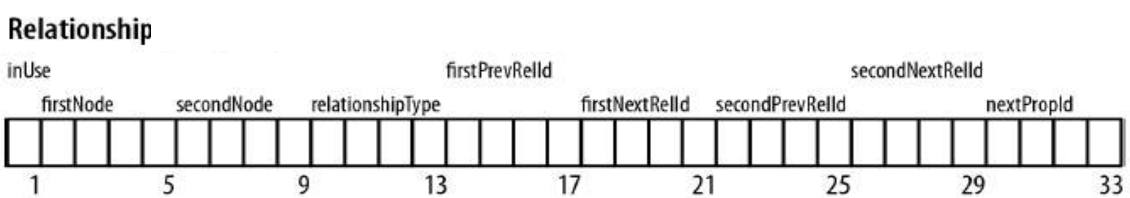
Order	
OrderID	**UserID**
1234	1
5678	1
...	...
5588	99

LineItem		
OrderID	**ProductID**	**Quantity**
1234	765	2
1234	987	1
...	...	...
5588	765	1

Product		
ProductID	**Description**	**Handling**
321	strawberry ice cream	freezer
765	potatoes	
...	...	
987	dried spaghetti	

Native Graph DB stores nodes and relationships directly, It makes retrieval efficient.



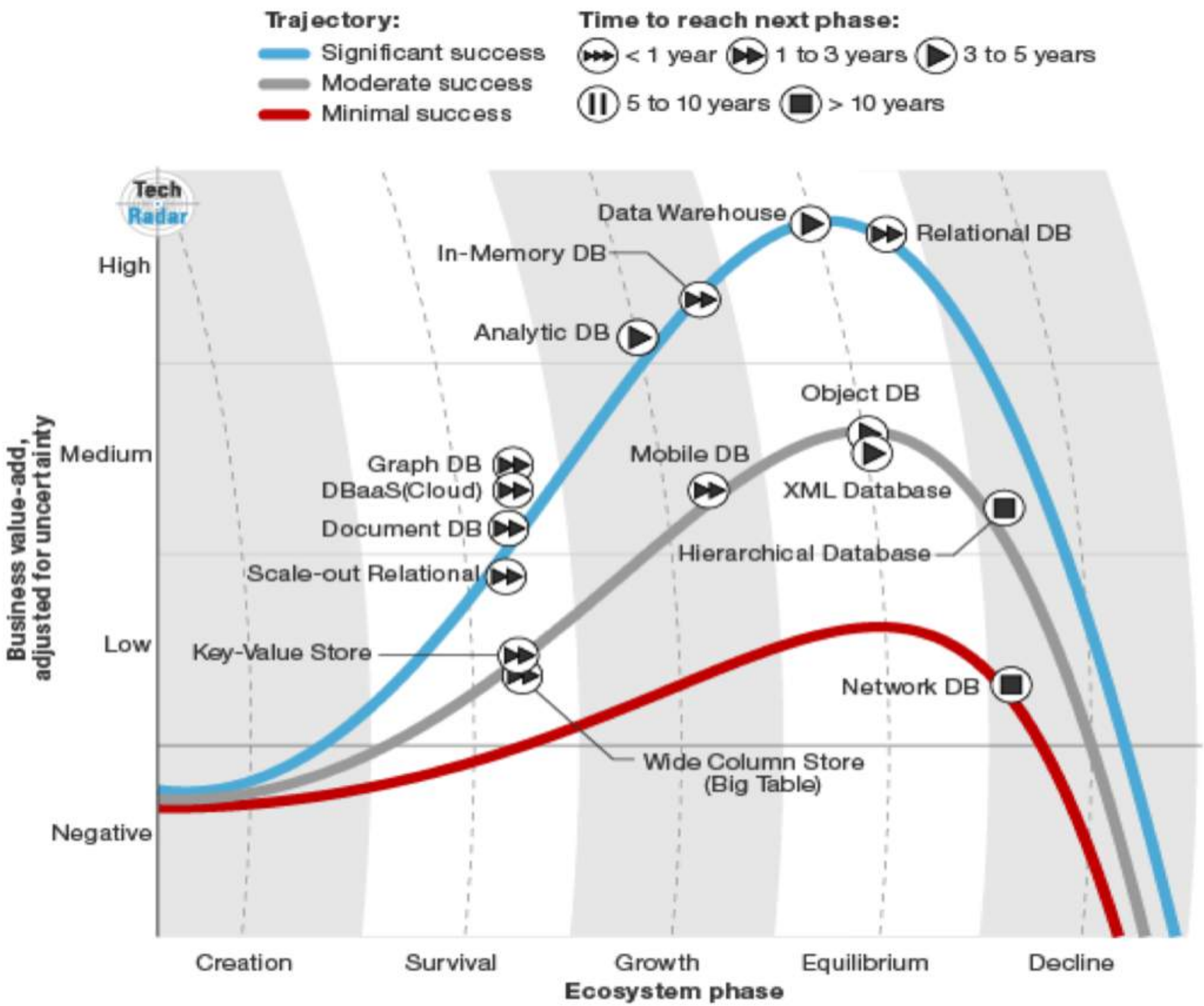


*Retrieving multi-step relationships is a '**graph traversal**' problem*

Technology ==> Top Layer: Graph, Bottom Layer: Graph

Cited “Graph Database” O'liey 2013

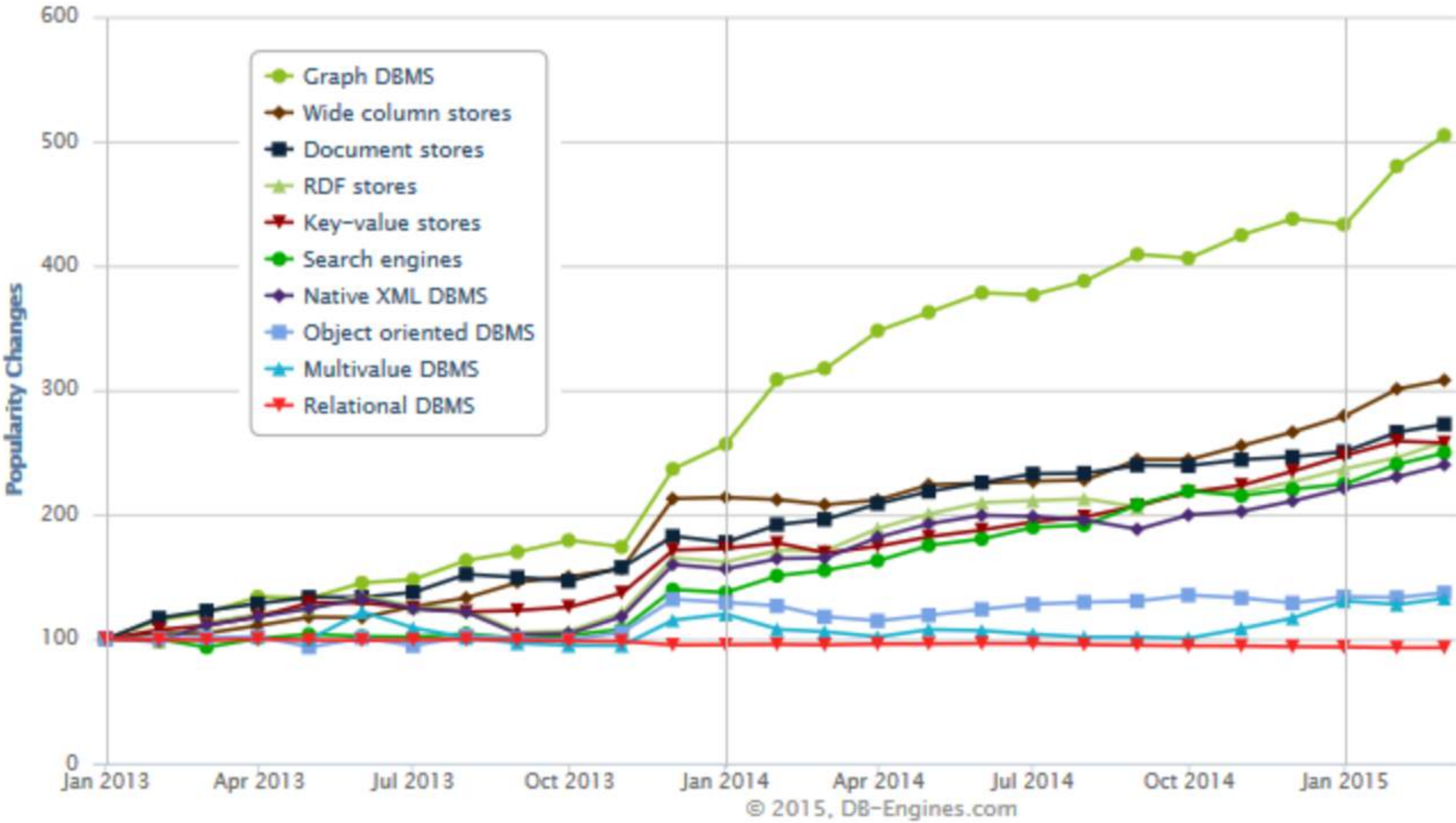
Forrester: Over 25% of enterprise will use Graph DB by 2017



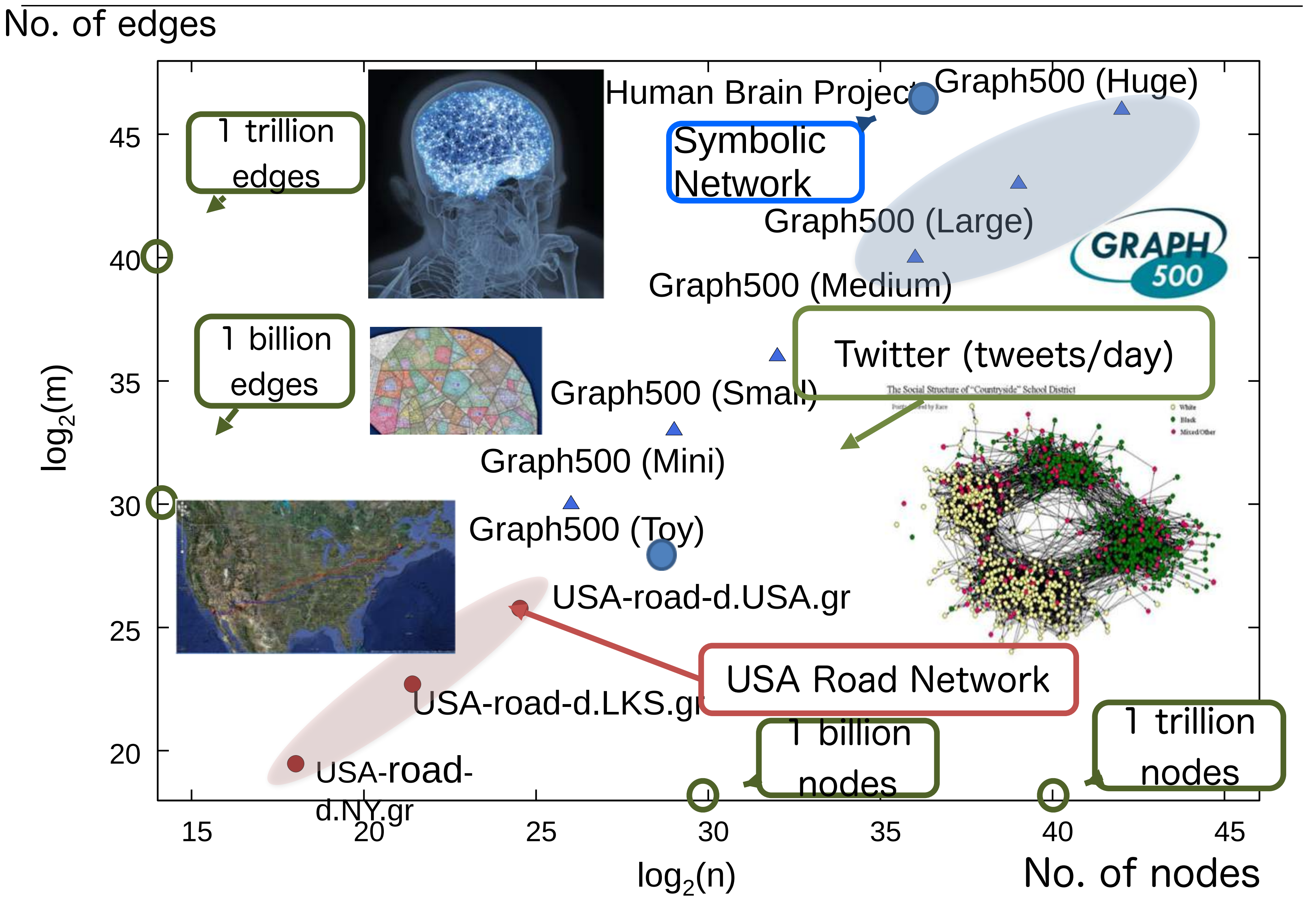
TechRadar: Enterprise DBMS, Q12014

Graph DB is in the significant success trajectory, and has the highest business value among the upcoming DBs.

GraphDB has the largest Popularity Change among DBMS lately

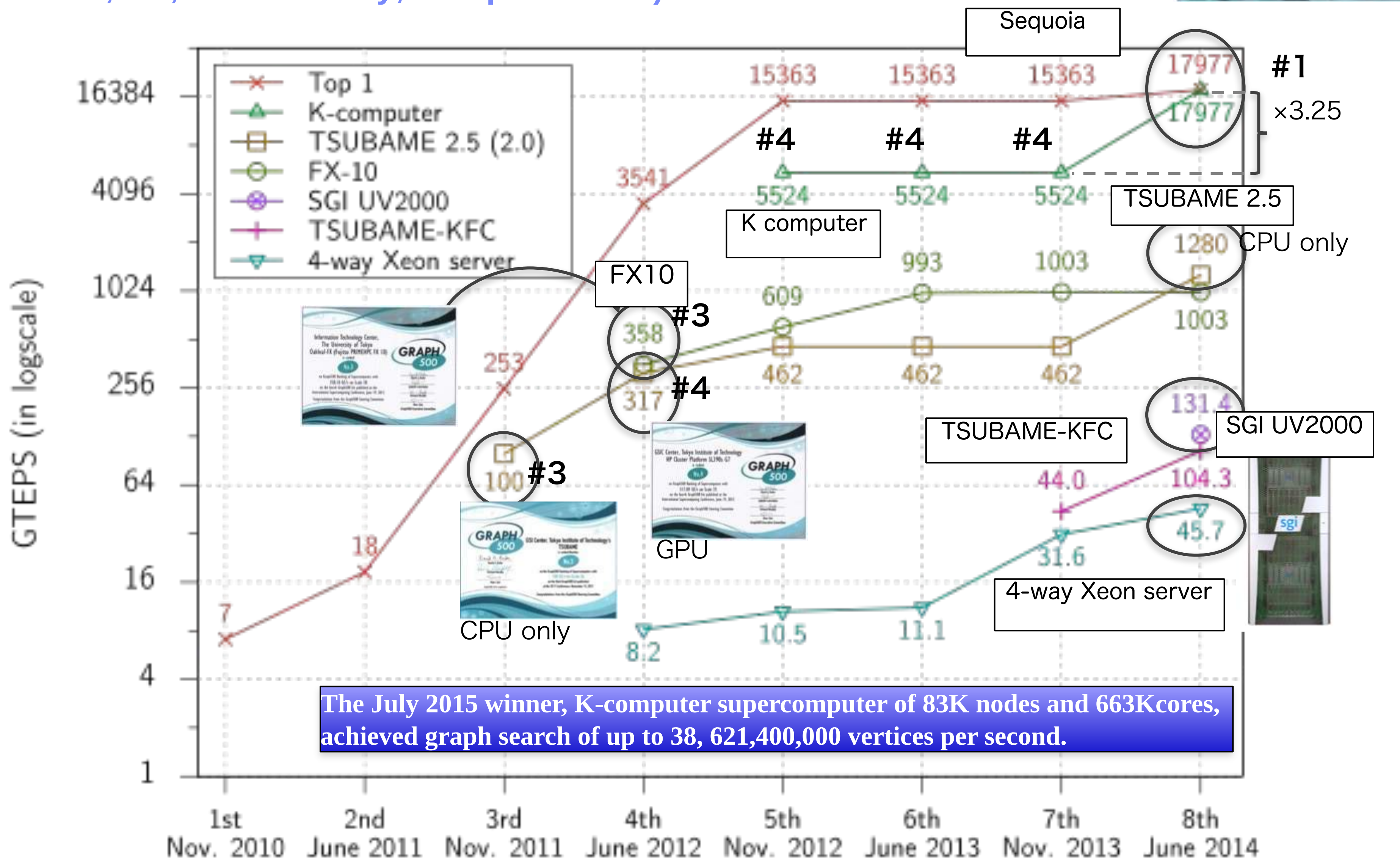


Comparison of graph size

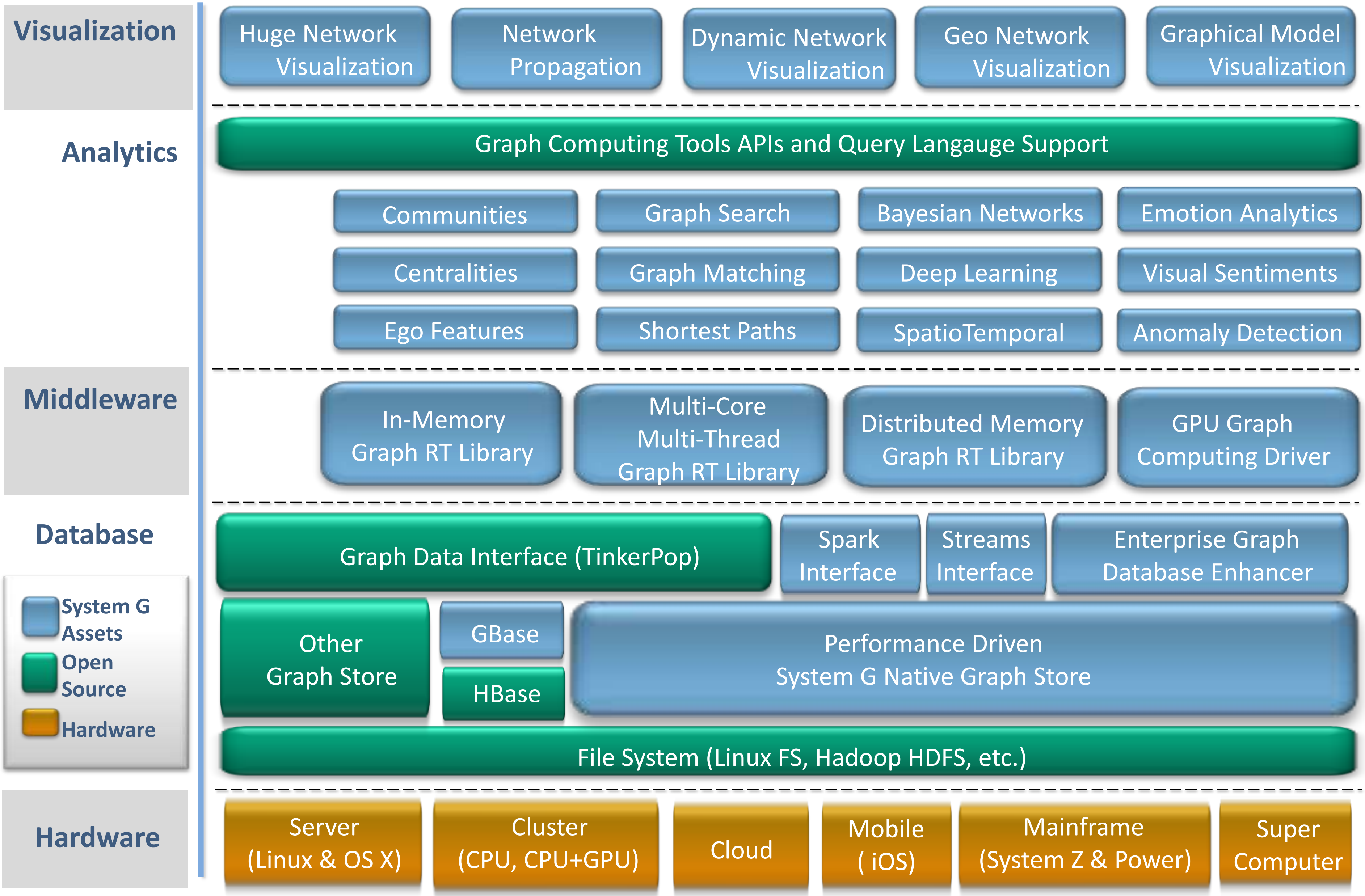


IBM System G Team

July 2015: IBM Research's Software powered all Top 3 winners of Graph 500 benchmark and 9 out of the Top 10 winners (supercomputers in US, Japan, France, UK, and Germany; except in China).



IBM System G Graph Computing Tools



IBM System G Application Use Cases

1. System G for Expertise Location
2. System G for Recommendation
3. System G for Commerce
4. System G for Financial Analysis
5. System G for Social Media Monitoring
6. System G for Telco Customer Analysis
7. System G for Watson
8. System G for Data Exploration and Visualization
9. System G for Personalized Search
10. System G for Anomaly Detection (Espionage, Sabotage, etc.)
11. System G for Fraud Detection
12. System G for Cybersecurity
13. System G for Sensor Monitoring (Smarter another Planet)
14. System G for Celluar Network Monitoring
15. System G for Cloud Monitoring
16. System G for Code Life Cycle Management
17. System G for Traffic Navigation
18. System G for Image and Video Semantic Understanding
19. System G for Genomic Medicine
20. System G for Brain Network Analysis
21. System G for Data Curation
22. System G for Near Earth Object Analysis

IBM System G

System G Team

Value of Social Connections

BusinessWeek

SEARCH SITE

Advanced Search

TOP NEWS | BW MAGAZINE | INVESTING | ASIA | EUROPE | TECHNOLOGY | AUTOS | INNOVATION | SMALL BIZ | B-SCHOOLS

APRIL 10, 2009

Insider Newsletter

A weekly summary of the best in BusinessWeek and BusinessWeek.com

NEWS THIS WEEK'S TOP STORY

Putting a Price on Social Connections

Researchers at IBM and MIT have found that certain e-mail patterns at work correlate with higher revenue production

EDITOR'S MEMO

Researchers at IBM and Massachusetts Institute of Technology say indulging in certain types of electronic communications makes for h productivity at work. Our Insider Top Story this week, "Putting a Price Connections," is part of our Special Report, The Value of Virtual Friends. Find out what sort of networking increases your value as an employee.

You've seen the advertisements on TV and in the paper—apparently there has never been a better time to sell Grandma's tiara. But hold on: Click through our slide show, "Selling Your Jewelry," and read the accompanying story first to see what you can get for what you've got. Have you ever had the thing appraised?

©Joern Pollex/Getty Images

IBM

According to happi thousands of IBM e were the better the $948 in revenue. IB introductions to inc

Value of Social Network

IBM

15,000 contributors in 76 countries; 92,000 unique IBM users

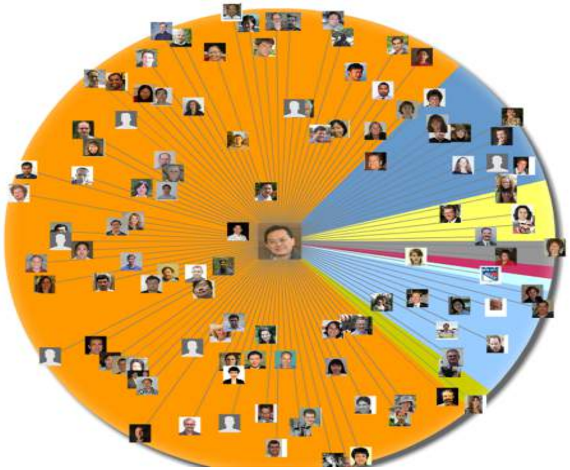
25,000,000 emails & SameTime messages (incl. Content features)

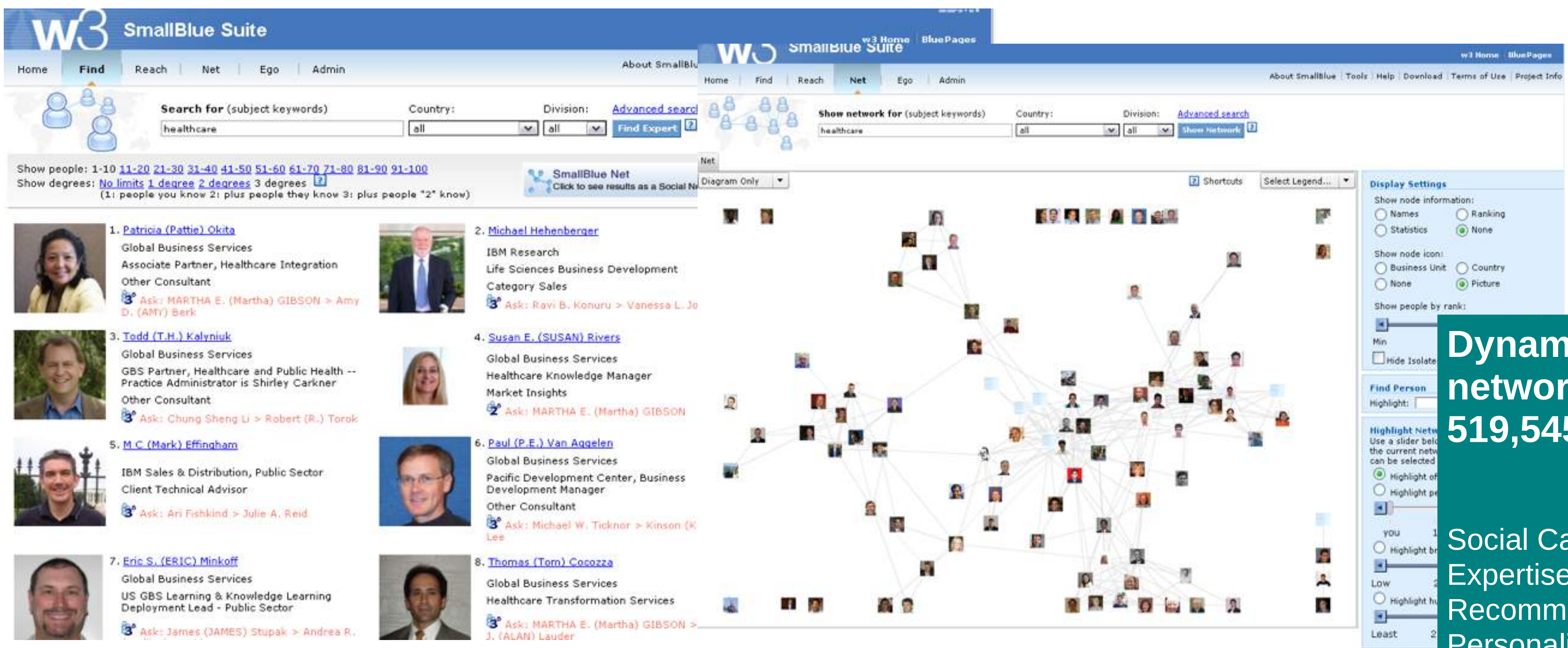
1,000,000 Learning clicks; 14M KnowledgeView, SalesOne, …, access data

1,000,000 Lotus Connections (blogs, flie sharing, bookmark) data

200,000 people's consulting financial databases

400,000 organization/demographic data



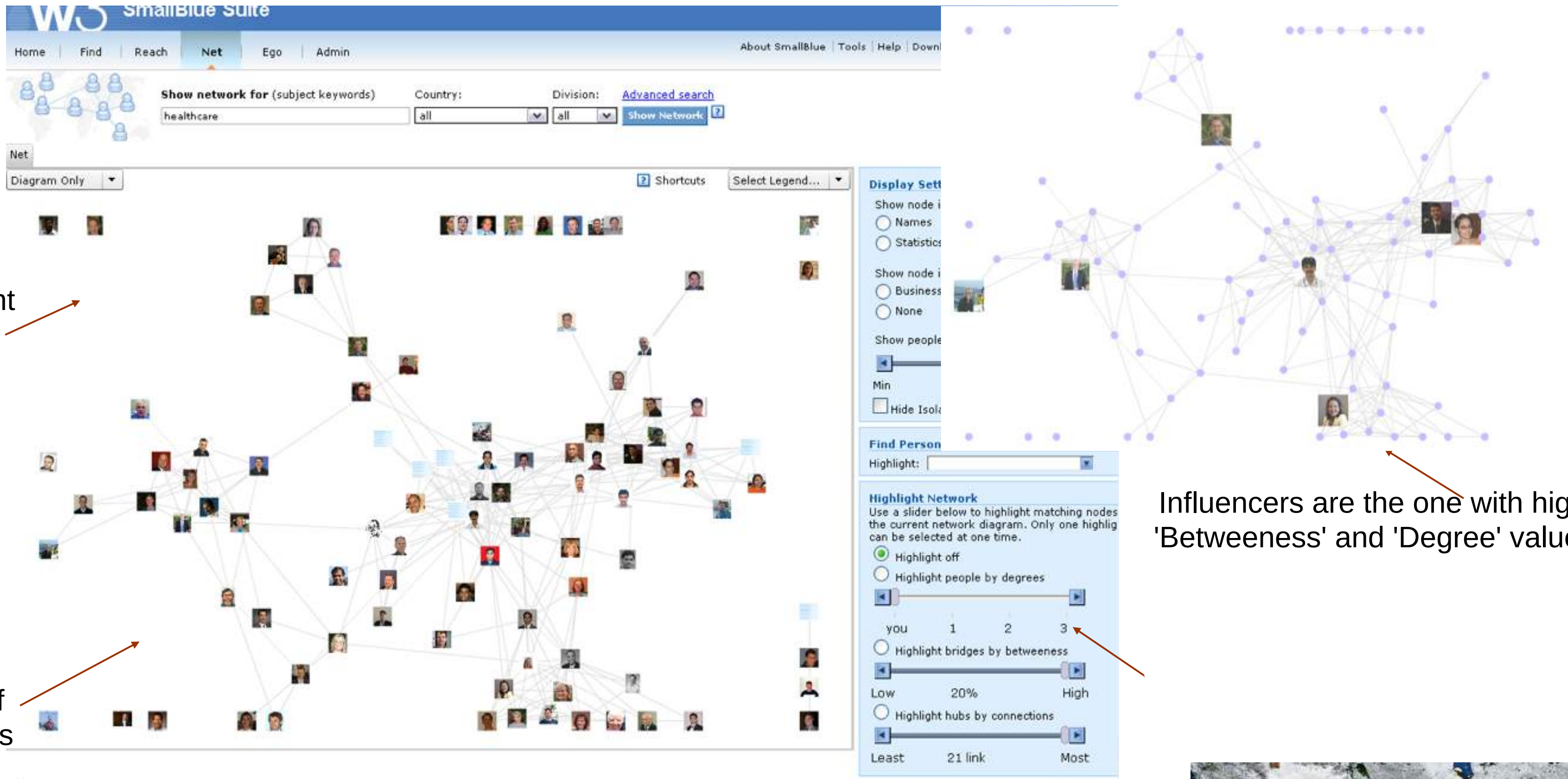


Dynamic networks of 519,545 IBMers

Social Capital
Expertise Location
Recommendation
Personalized Search

Finding Influencer and Ranking Expertise – Social Network Analysis

- Decades of Social Science studies demonstrates that (social) network structure is the key indicator determining a person's influence, organizational operation efficiency, social capital to get help, potential to be successful, etc.
- Who are the key bridges? Who have the most connections? How do these experts cluster?
- Analogy – Google founders utilized the concept of network analysis on webpages to create ranking.



Independent experts on healthcare

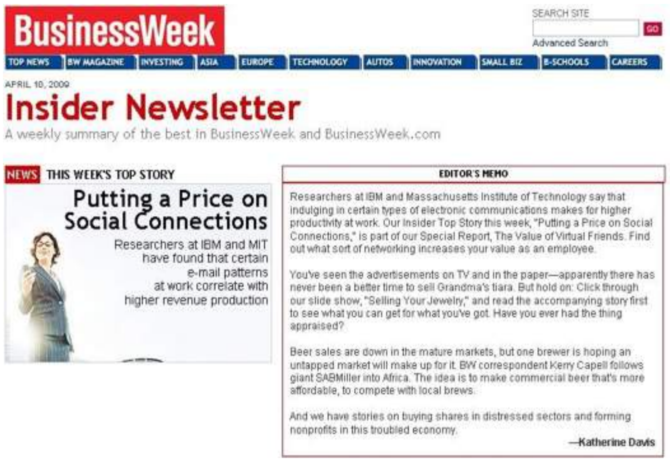
A cluster of XYZ experts

Influencers are the one with high 'Betweeness' and 'Degree' values

SmallBlue analyzes underlining dynamic network structure in enterprise

519,545 IBMer Network on May 9, 2012

Network Value Analysis – First Large-Scale Economical Social Network Study



BusinessWeek

SEARCH SITE GO Advanced Search

TOP NEWS | BW MAGAZINE | INVESTING | ASIA | EUROPE | TECHNOLOGY | AUTOS | INNOVATION | SMALL BIZ | B-SCHOOLS | CAREERS

APRIL 10, 2009

Insider Newsletter

A weekly summary of the best in BusinessWeek and BusinessWeek.com

NEWS THIS WEEK'S TOP STORY

Putting a Price on Social Connections

Researchers at IBM and MIT have found that certain e-mail patterns at work correlate with higher revenue production

EDITOR'S MEMO

Researchers at IBM and Massachusetts Institute of Technology say that indulging in certain types of electronic communications makes for higher productivity at work. Our Insider Top Story this week, "Putting a Price on Social Connections," is part of our Special Report, The Value of Virtual Friends. Find out what sort of networking increases your value as an employee.

You've seen the advertisements on TV and in the paper—apparently there has never been a better time to sell Grandma's tiara. But hold on: Click through our slide show, "Selling Your Jewelry," and read the accompanying story first to see what you can get for what you've got. Have you ever had the thing appraised?

Beer sales are down in the mature markets, but one brewer is hoping an untapped market will make up for it. BW correspondent Kerry Capell follows giant SABMiller into Africa. The idea is to make commercial beer that's more affordable, to compete with local brews.

And we have stories on buying shares in distressed sectors and forming nonprofits in this troubled economy.

—Katherine Davis

<u>Productivity effect from network variables</u>

- An additional person in network size ~ $948 revenue per year
- Each person that can be reached in 3 steps ~ $0.163 in revenue per month
- A link to manager ~ $1074 in revenue per month
- 1 standard deviation of network diversity (1 - constraint) ~ $758
- 1 standard deviation of btw ~ -$300K
- 1 strong link ~ $-7.9 per month

- Structural Diverse networks with abundance of structural holes are associated with higher performance.
  - *Having diverse friends helps.*
- Betweenness is negatively correlated to people but highly positive correlated to projects.
  - *Being a bridge between a lot of people is bottleneck.*
  - *Being a bridge of a lot of projects is good.*
- Network reach are highly corrected.
  - *The number of people reachable in 3 steps is positively correlated with higher performance.*
- Having too many strong links — the same set of people one communicates frequently is negatively correlated with performance.
  - *Perhaps frequent communication to the same person may imply redundant information exchange.*

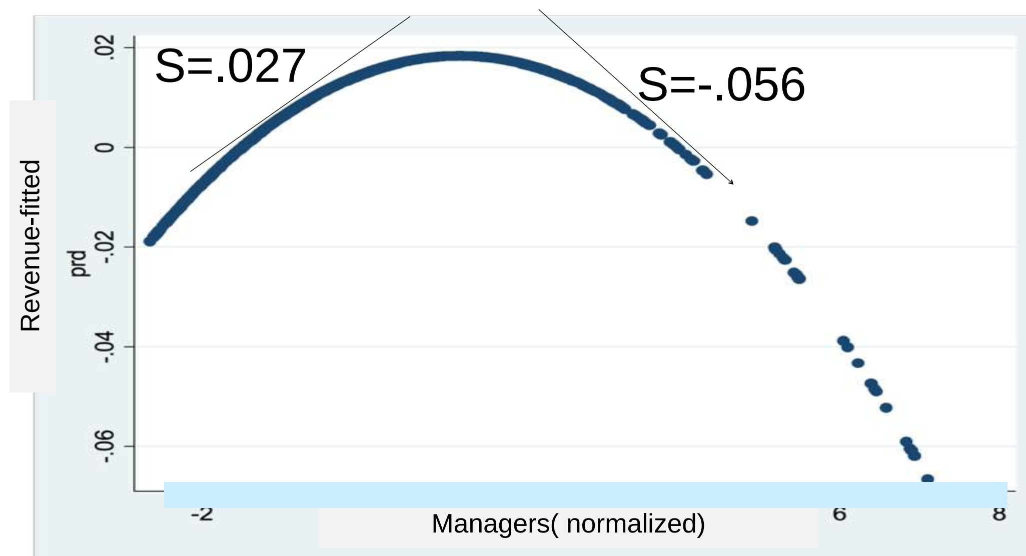
Project Team Composition—Managers

The number of managers in a project exhibit an inverted-U shaped curve.

1. Having managers in a project is correlated with team performance initially.
2. Too many managers in a project is negatively associated with team performance.

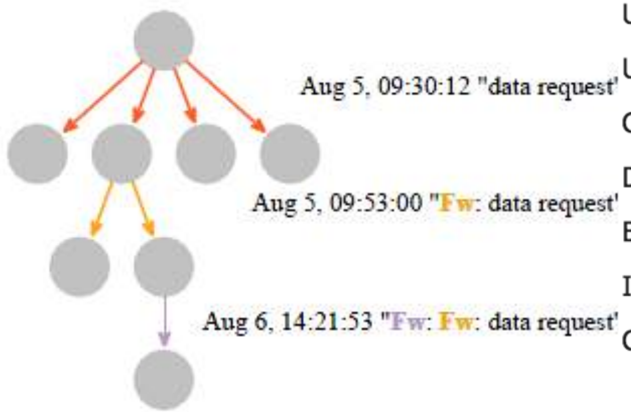
$$revenue = \alpha + \beta_1 \cdot mgr + \beta_2 \cdot mgr^2 + \gamma_1 \cdot otherfactor_1 + \ldots + \gamma_k \cdot otherfactor_k + \varepsilon$$

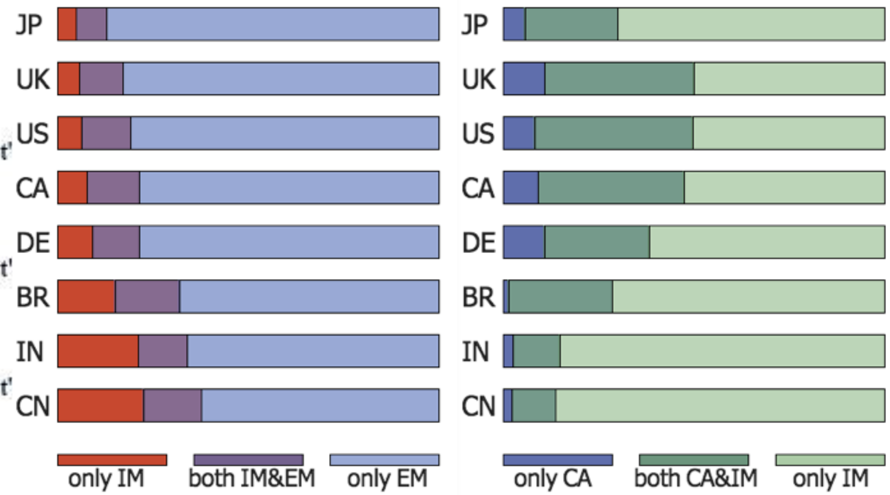
# Managers in project	β_1	2733.9*** (537.5)
(# Managers in project) ^2	β_2	-682.02*** (215.3)



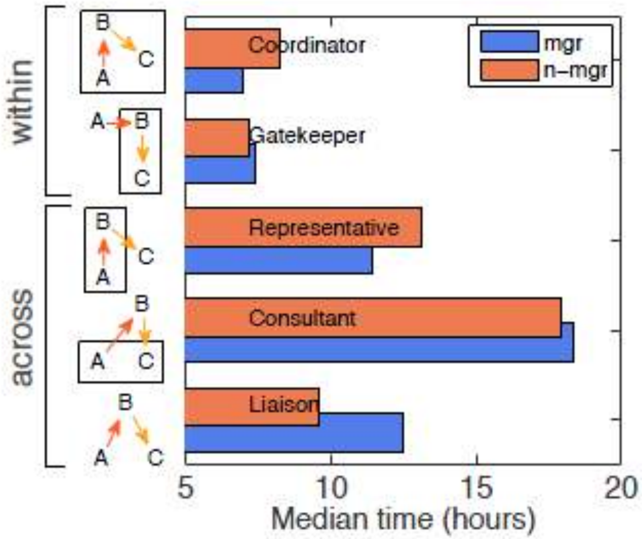
21

Behavior depends on Culture and Roles

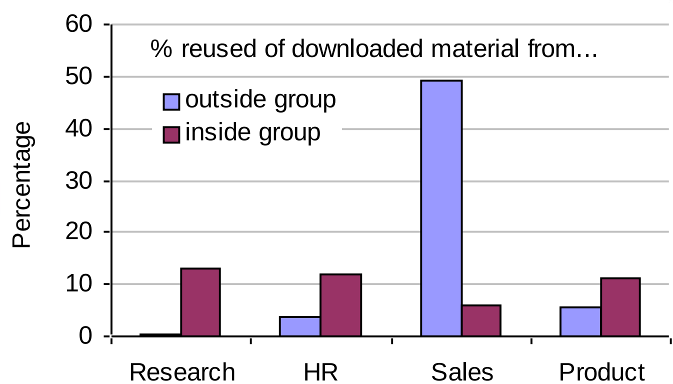




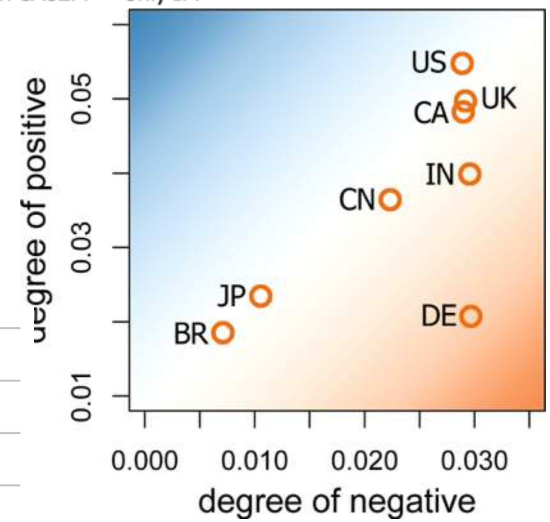
Culture difference of normal behavior (ICIS 2011)



Role difference of normal behavior



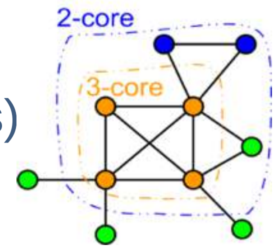
[Mejova, CHI 2011]



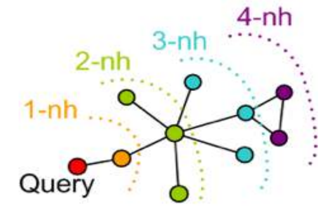
Organization difference of normal behavior

System G Graph Analytical Tools

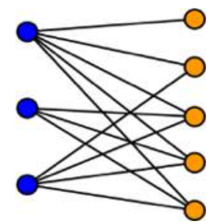
- ***Network topological analysis*** tools
  - Centralities (degree, closeness, betweenness)
  - PageRank
  - Communities (connected component, K-core, triangle count, clustering coefficient)
  - Neighborhood (egonet, K-neighborhood)
- ***Graph matching and search*** tools
  - Graph search/filter by label, vertex/edge properties (including geo locations)
  - Graph matching
  - Collaborative filtering
- ***Graph path and flow*** tools
  - Shortest paths
  - Top K-shortest paths
- ***Probabilistic graphical model*** tools
  - Bayesian network inference
  - Deep learning

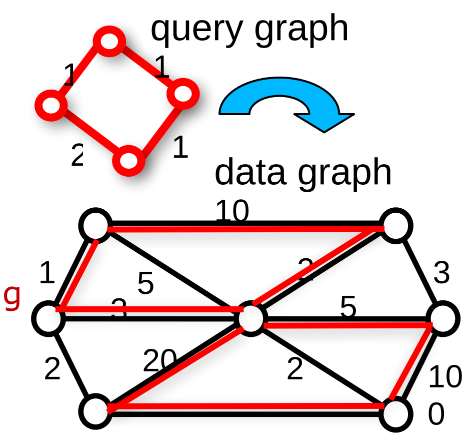


K-core

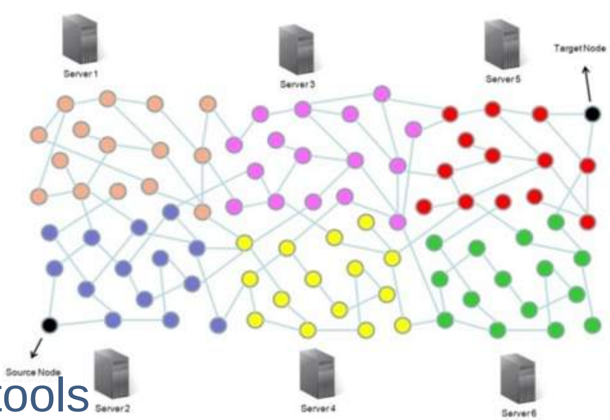


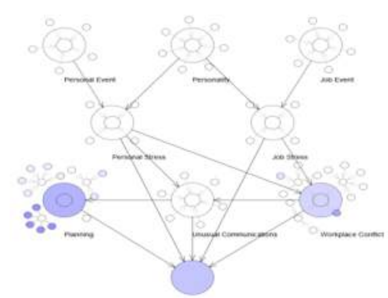
K-neighborhood


Collaborative filtering
Bipartite weighted graph matching



Graph matching


Top k-shortest paths


Bayesian network inference

Characteristics of IBM System G Graph Analytics

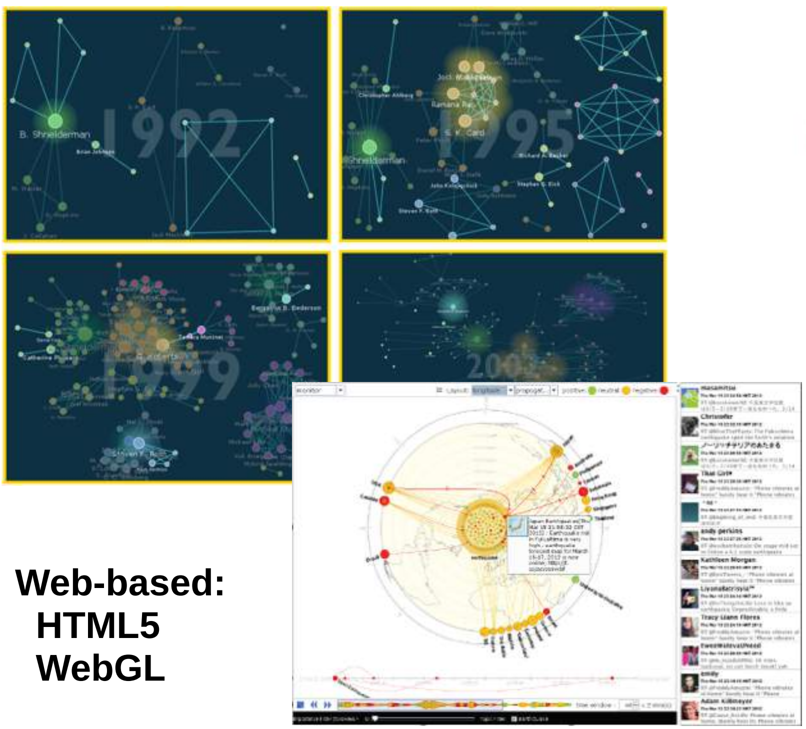
- Cover a wide range of graph analytics to support many application use cases in different domains, e.g.:
    - Enterprise social network analysis, expertise search, knowledge recommendation
    - Financial/security anomaly/fraud detection
    - Social media monitoring and analysis
    - Cellular network analytics in Telco operation
    - Patient and disease analytics for healthcare
    - Live neural brain network analysis
- Provide efficient in-memory computation as well as on-disk persistence
- Optimal performance enabled by IBM System G graph database technologies that focus on efficient use of available computing resources with architecture-aware design to leverage system/architecture advantages
- Single-threaded, concurrent (shared memory), and distributed versions
- Multiple deployment options to suit different customer preferences and needs
    - C++ executables in Linux environments (Redhat CentOS, Ubuntu, Mac OS X, Power)
    - TinkerPop (Blueprints) API
    - gShell (a shell-like environment with interactive, batch, and server/client modes to operate multiple graph stores simultaneously)
    - Gremlin console
    - REST API Web service
    - Python wrapper

IBM System G

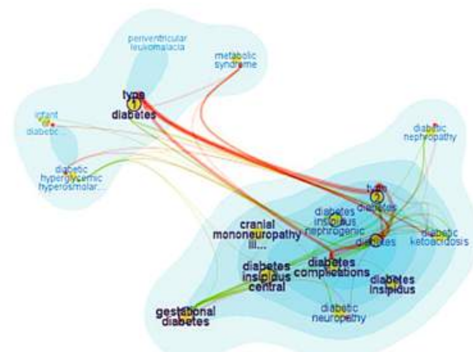
Existing foundation of 16 types of graph visualization assets in these 4 categories:

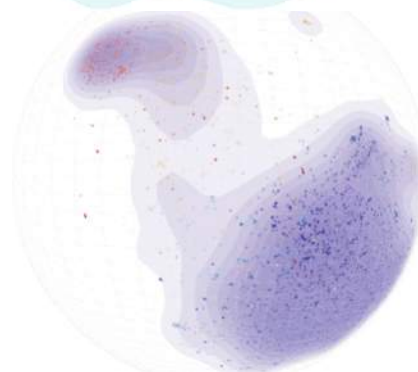
- **Multivariate Graphs**: nodes and edges have multivariate attributes. E.g., healthcare graphs, workflow graphs, behavior reasoning graphs, etc.
- **Heterogeneous Graphs**: graphs in which nodes and edges are in different categories and types. E.g.: bipartite/tripartite/multi-partite graphs, geospatial graphs, etc.
- **Dynamic Graphs**: graphs whose topology and attributes change over time. E.g., relationship graphs, information propagation graphs, etc.
- **Big Graphs**: graphs with millions or even billions of nodes and edges. Hierarchical-based visualization or infinite-plane based visualization. E.g., social graphs, knowledge graphs, etc.

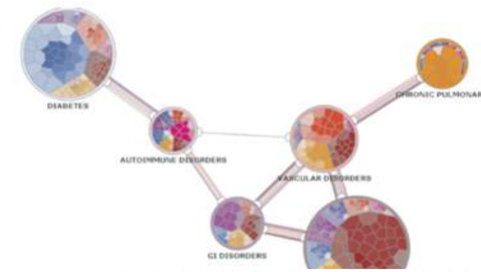
http://systemg.ibm.com

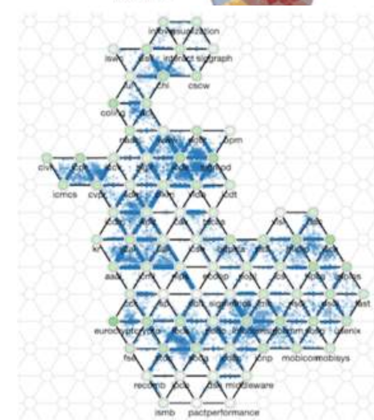


Web-based:
HTML5
WebGL









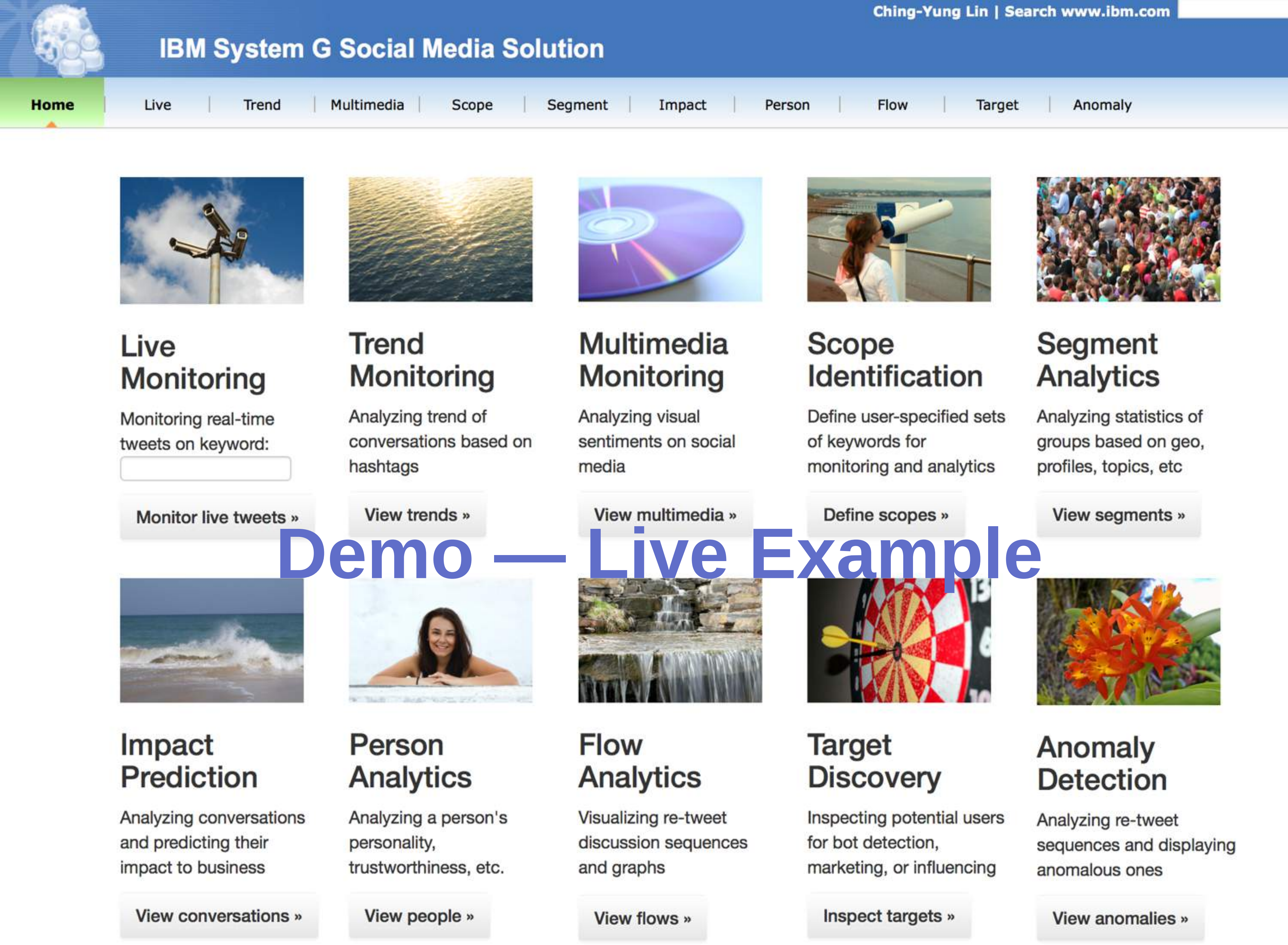
Demo — Exploration

Social Media Monitoring

Modeling, Tracking and Affecting Information Dissemination in Context

==> 26 Fundamental Research Tasks organized in 3 Thrusts — Modeling, Tracking and Affecting

Demo — Live Example



Anomaly Detection at Multiple Scales (ADAMS) Summary

IBM

A novel **Cognitive Security System** to Detect and Predict Abnormal Behaviors in Organization from large-scale multimodality data of people through graph computing, cognitive analytics, data mining, and machine learning.

Emails
Instant Messaging
Web Access
Executed Processes
Printing
Copying
Log On/Off

→

Analytics Infrastructure
Data sensing
Data processing
Data storage

→

Graph analysis
Behavior analysis
Content analysis
Emotion analysis

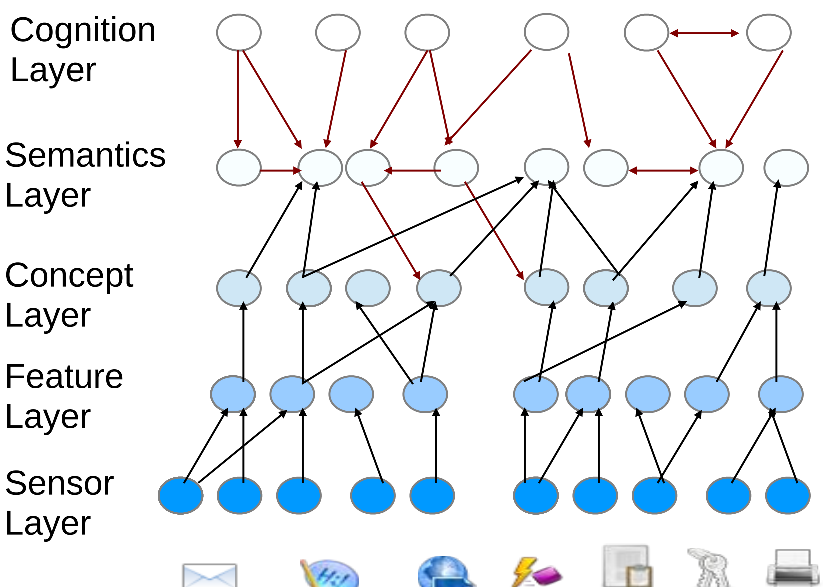
→

Cognitive Reasoning

→

Detection, Prediction, and Exploration Interface

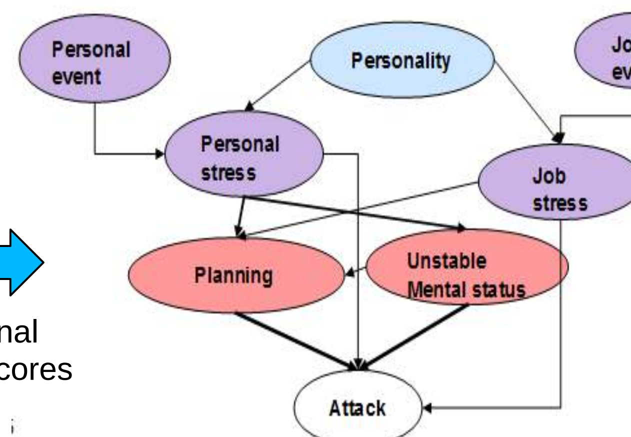
Infrastructure + ~ 70 Types of Analytics (489 Detectors) + Reasoning + User Interface



Available existing data

●: observations ○: hidden states

~500 original detector scores / per day / per person



Espionage Detection
Sabotage Detection
Fraud Detection

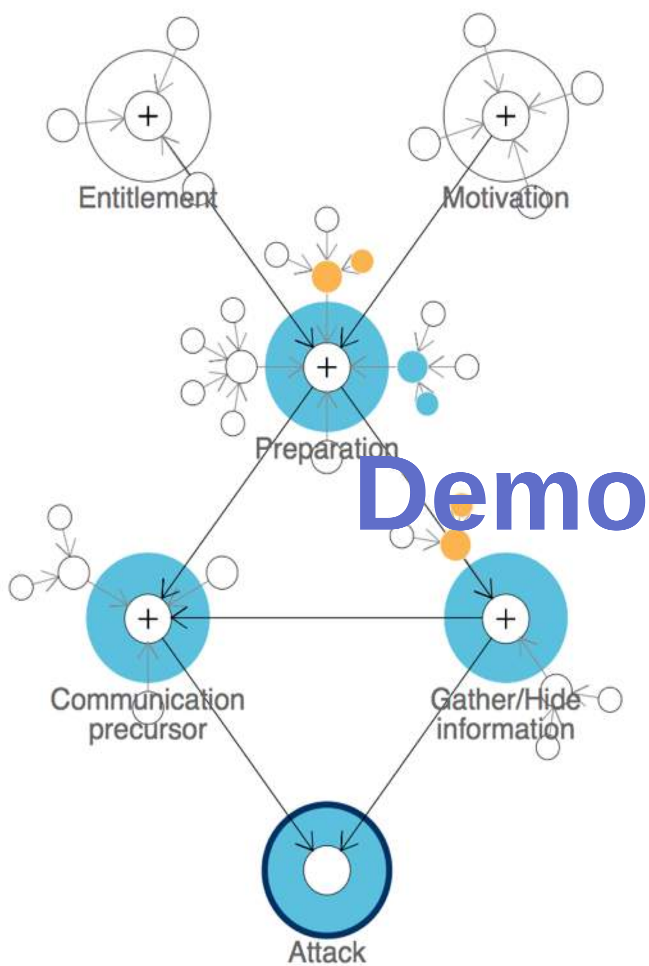
3 potential 'Attack' scores per person

IBM System G Insider Threat Solution

Threat: Espionage   User: ADP3667

Period: Jun 2014 1 to Jun 2014 30

Date: 2014-06-18



Espionage: Attack

Timeline for user ADP3667 from 2014-06-01 to 2014-06-30

Date	Attack
2014-06-01	
2014-06-02	
2014-06-03	
2014-06-04	
2014-06-05	
2014-06-06	
2014-06-07	
2014-06-08	
2014-06-09	
2014-06-10	
2014-06-11	
2014-06-12	
2014-06-13	
2014-06-14	
2014-06-15	
2014-06-16	
2014-06-17	
2014-06-18	50
2014-06-19	
2014-06-20	
2014-06-21	
2014-06-22	
2014-06-23	
2014-06-24	30.15
2014-06-25	30.15

Demo — Reasoning

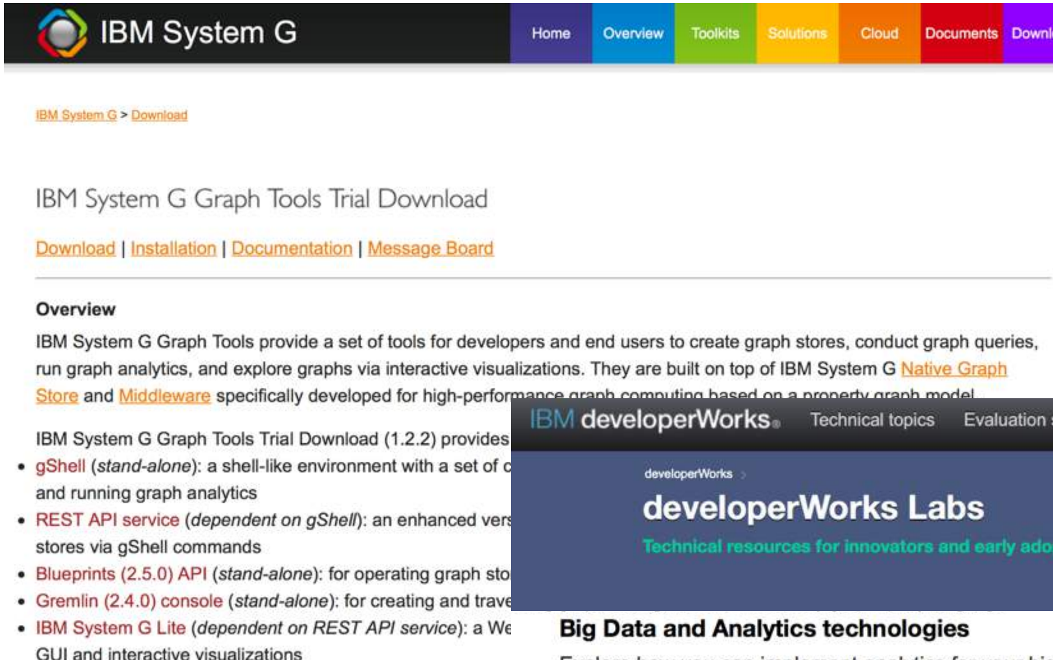
Fraud

Sabotage

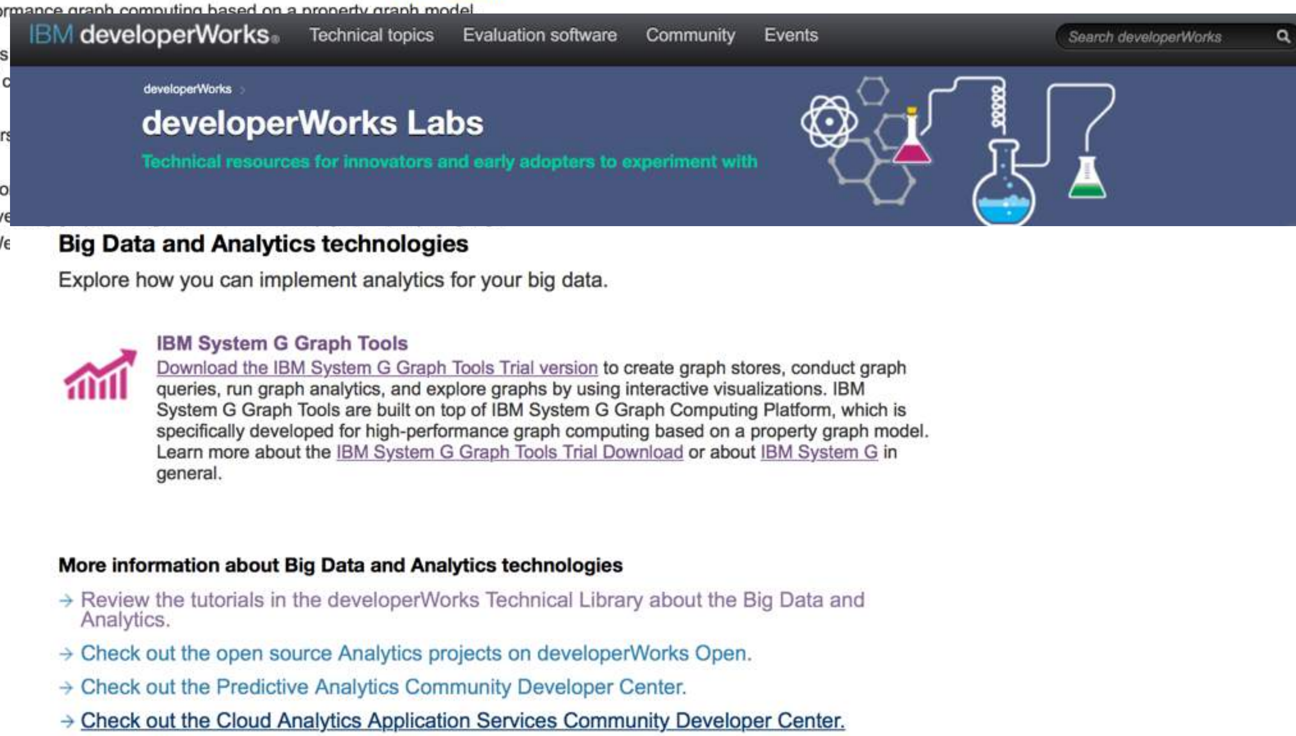
Espionage

Download IBM System G Standard Edition (on-premise)

http://systemg.research.ibm.com/download.html



or
http://www.ibm.com/developerworks/labs/



Resources

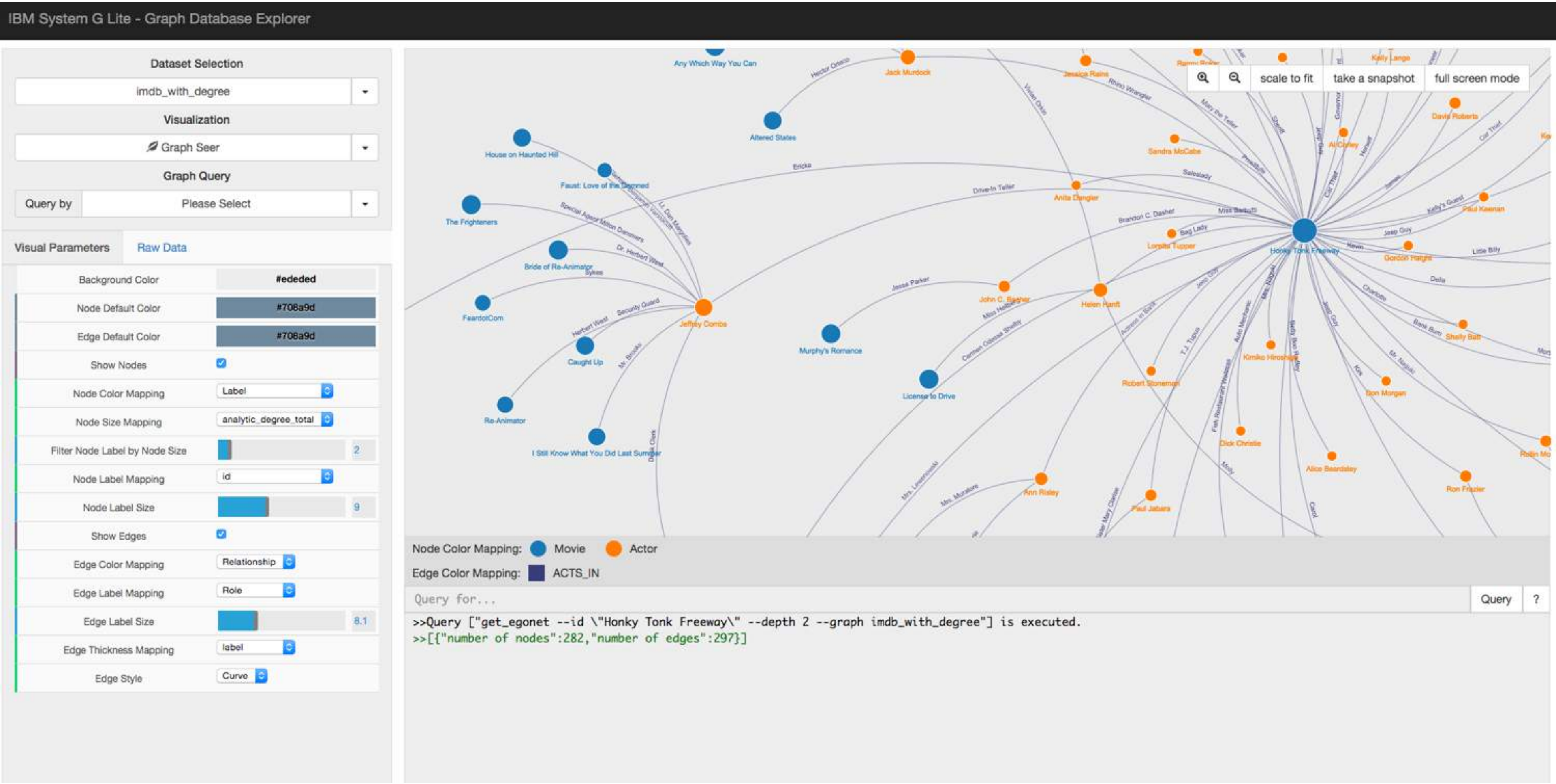
- IBM System G on Bluemix (need registration)
  - http://systemg.mybluemix.net
- IBM System G Graph Analytics Overview
  - http://systemg.research.ibm.com/analytics.html
- IBM System G Graph Tools Trial Download
  - http://systemg.research.ibm.com/download.html
- IBM System G Graph Tools Installation Guide and Documentation
  - http://systemg.research.ibm.com/setup.html

IBM System Visualizer (SystemG-Lite)

IBM System G Visualizer – Graph Data Explorer

Visual Query Panel

Visualization Panel



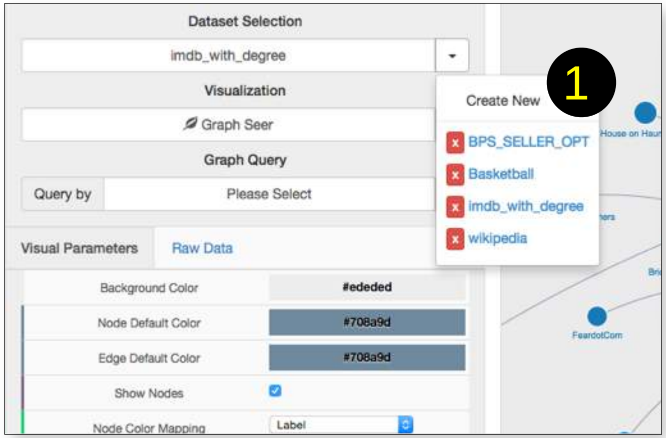
Visual Mapping Panel

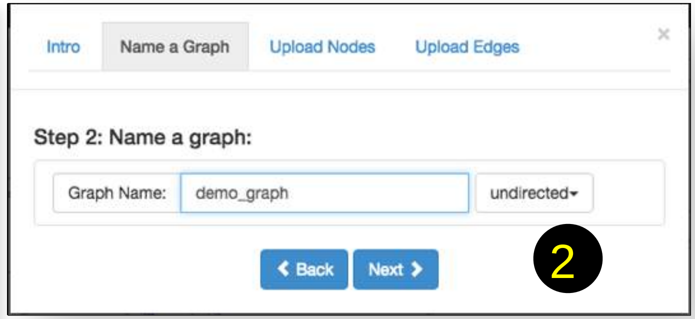
Console Panel

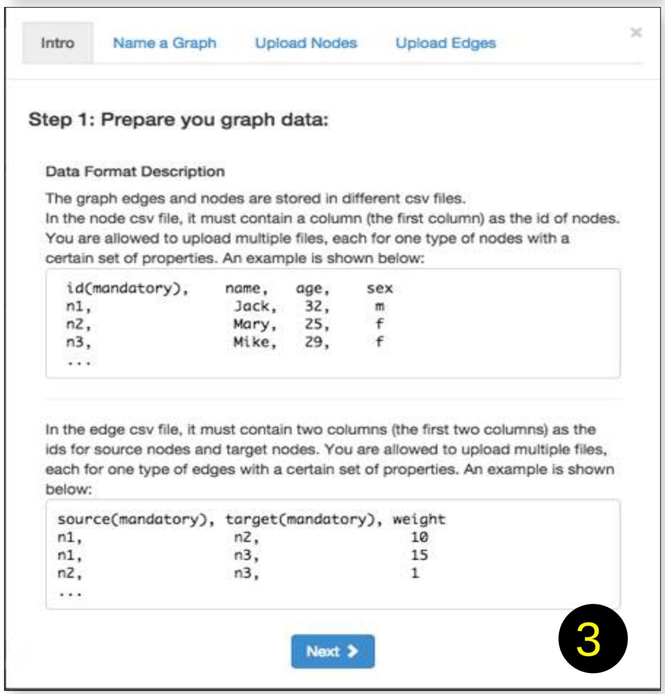
Panel Introduction

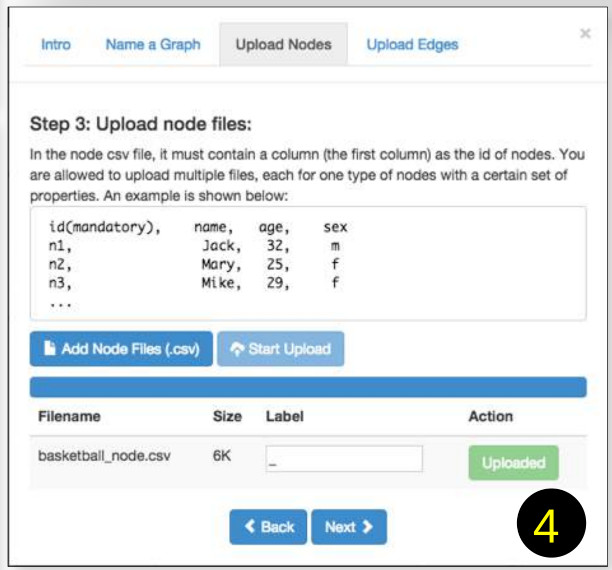
- Visual Query Panel
  - Providing users a friendly UI to create, delete, and query graphs from the System G native store.
- Console Panel
  - Display all the interaction information with System G native store.
  - Execute user defined query.
- Visualization Panel
  - Rendering graph structure on screen for users to visually explore graphs.
- Visual Mapping Panel
  - Customizing rendering effects to show desired graph information.

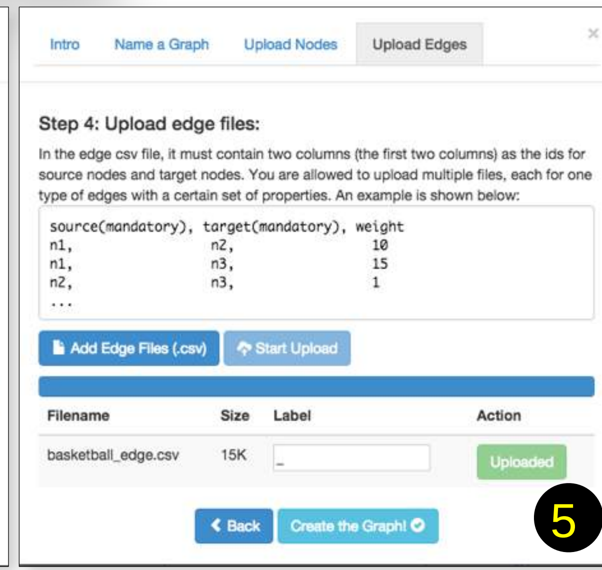
Visual Query Panel – Creating a graph





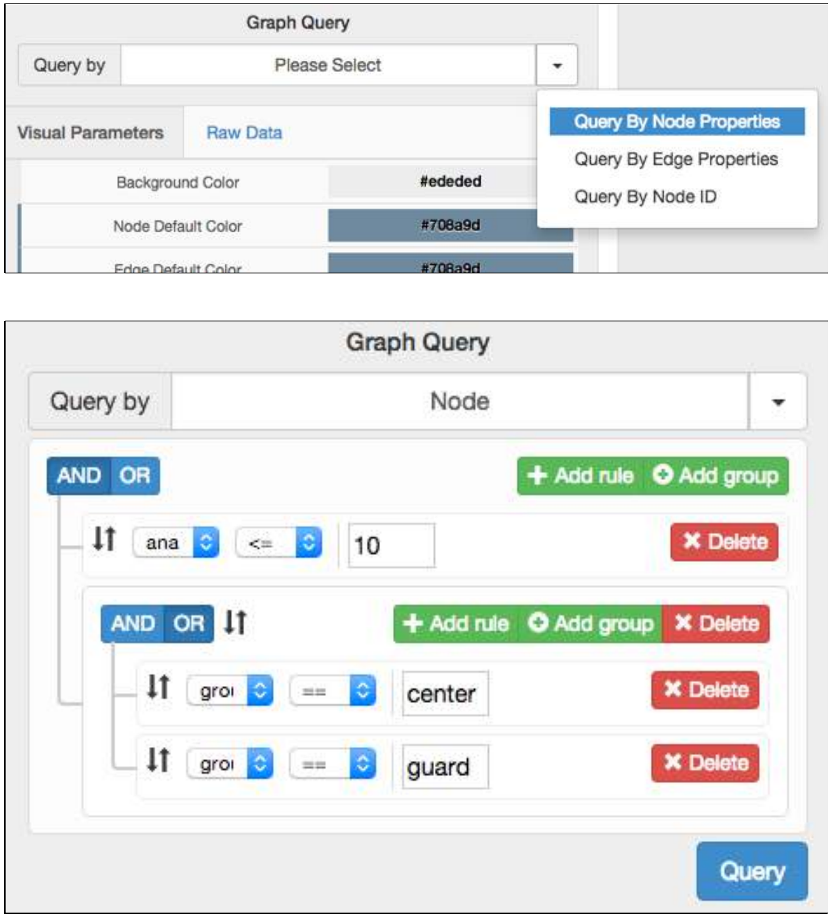


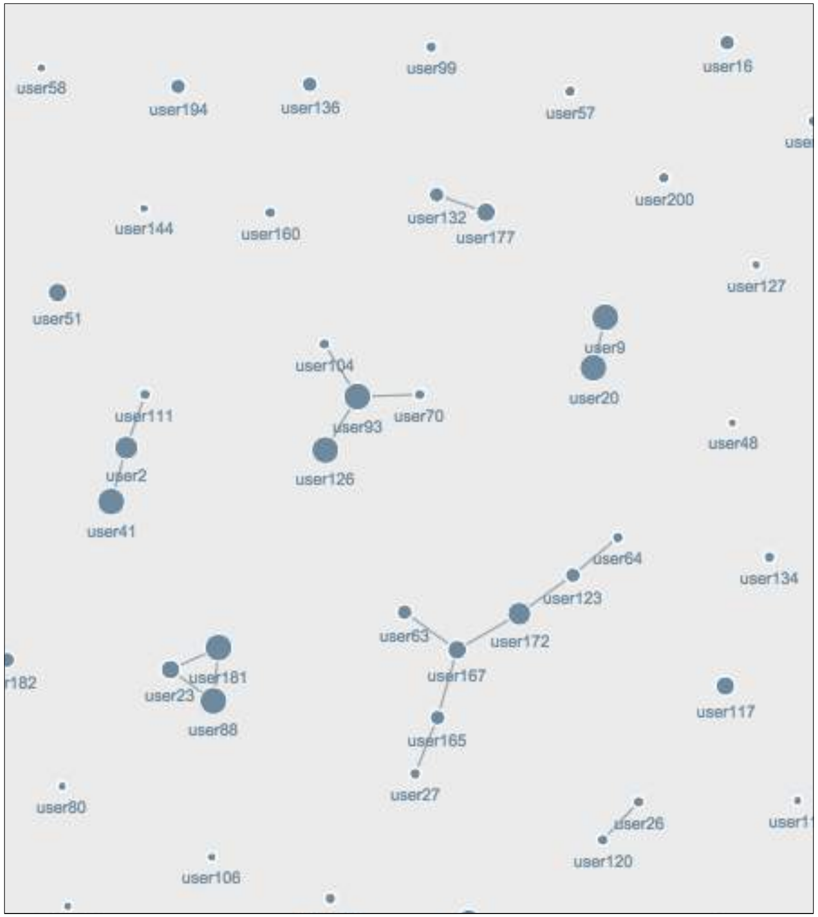




1: Click "Create Graph"; 2: Prepare the graph data
3: Set the graph name; 4: Upload node files;
5: Upload edge files and finalize creating the graph.

Visual Query Panel – Visual Query Builder





“analytics_degree <= 10 and (group == “center” or group == “guard”)

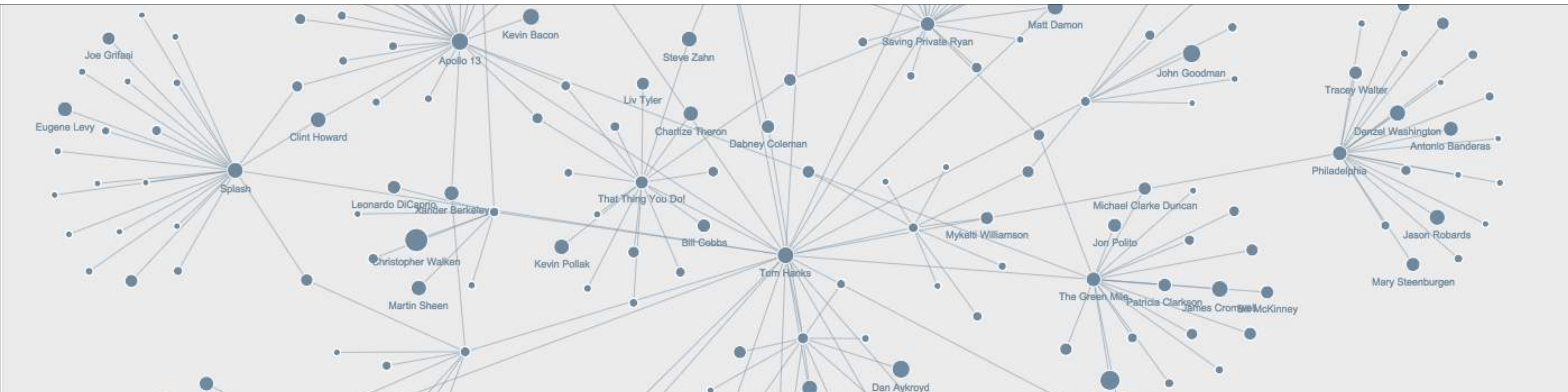
Console Panel – User typed query

```
find_vertex_max_degree --graph Basketball --edgetype all
```

Query ?

```
>>Query ["print_all --graph Basketball"] is executed.
>>[{"number of nodes":199,"number of edges":826}]
>>Query ["find_vertex_max_degree --edgetype all --graph Basketball"] is executed.
>>[{"vertex id":"user72"},{"all-degree":46}]
```

Query with no graph returned



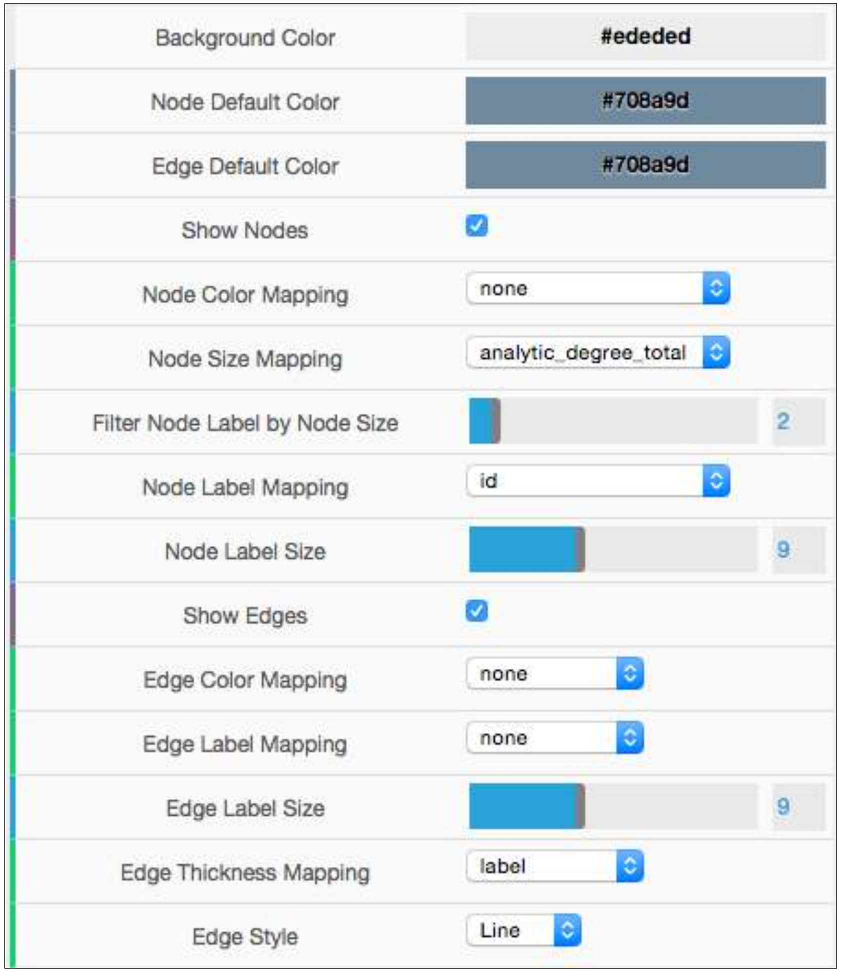

```
get_egonet --graph imdb_with_degree --id "Tom Hanks" --depth 2
```

Query ?

```
>>Query ["print_all --graph Basketball"] is executed.
>>[{"number of nodes":199,"number of edges":826}]
>>Query ["find_vertex_max_degree --edgetype all --graph Basketball"] is executed.
>>[{"vertex id":"user72"},{"all-degree":46}]
>>Query ["get_egonet --id \"Tom Hanks\" --depth 1 --graph imdb_with_degree"] is executed.
>>[{"number of nodes":26,"number of edges":25}]
>>Query ["get_egonet --id \"Tom Hanks\" --depth 2 --graph imdb_with_degree"] is executed.
>>[{"number of nodes":383,"number of edges":401}]
```

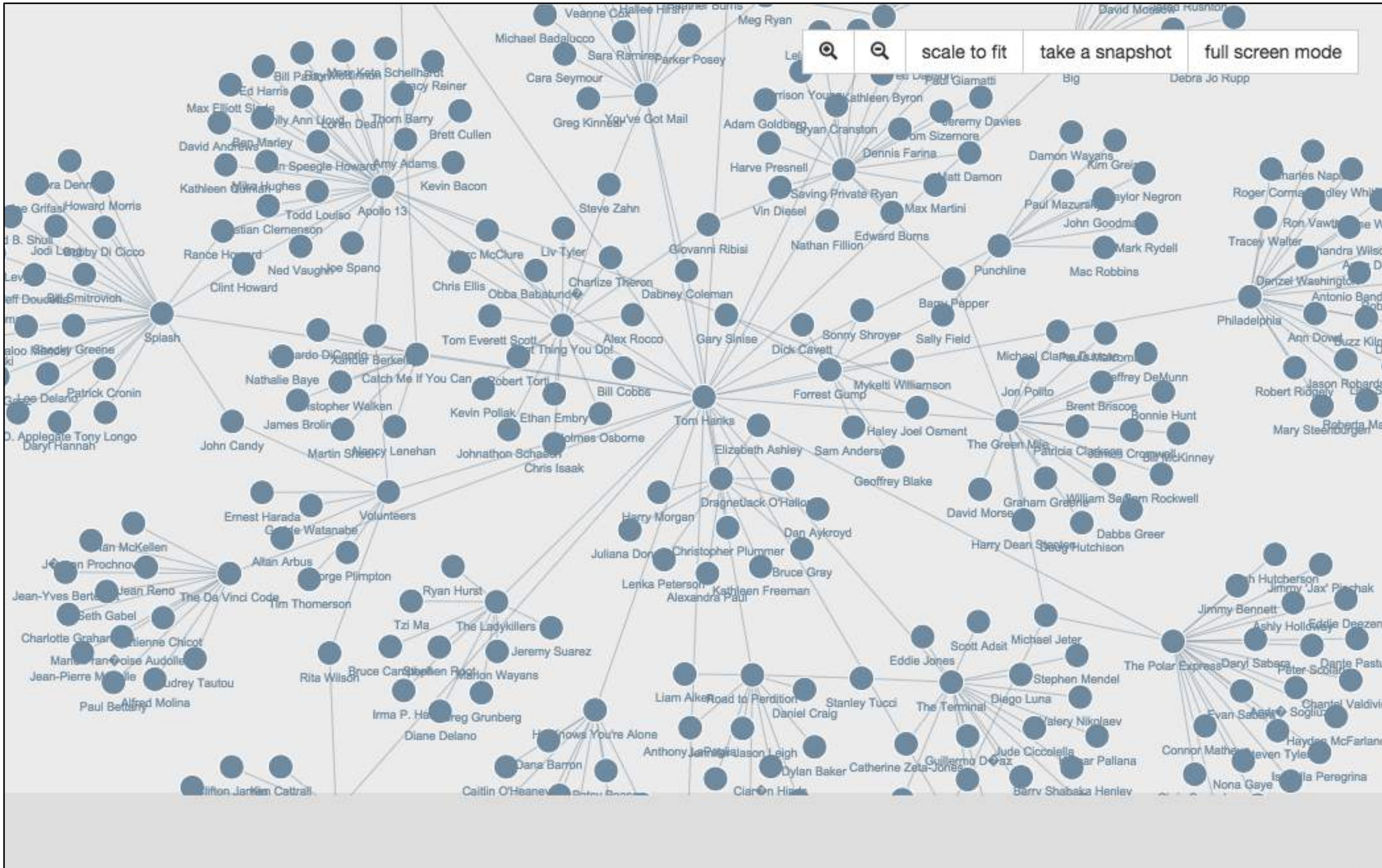
Query with graph returned

Visual Mapping Panel

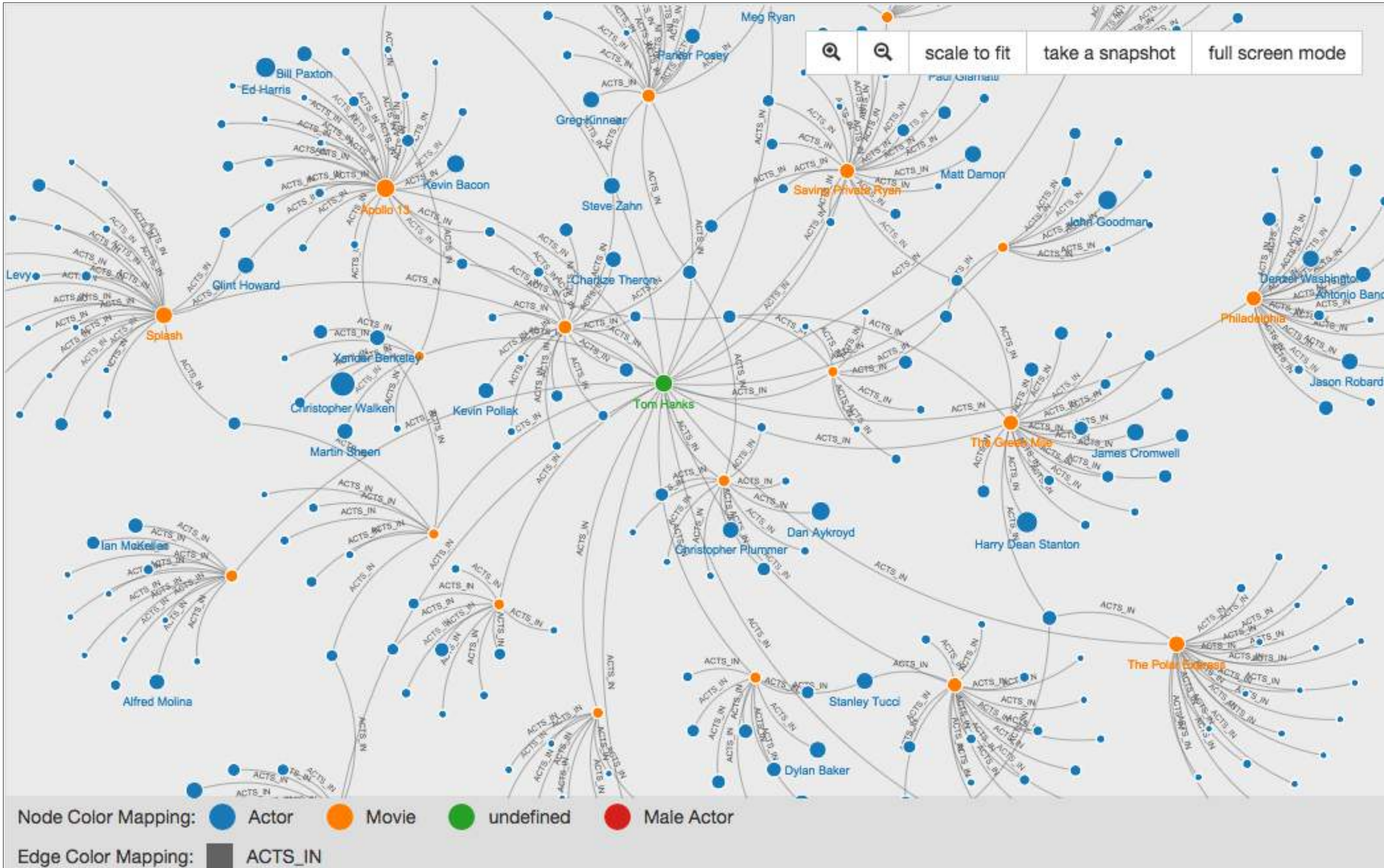


Name	Functionality
Background Color	Change the background color of the canvas.
Node Default Color	Set a unified color for all nodes.
Edge Default Color	Set a unified color for all edges.
Show Nodes	Set the visibility of all nodes.
Node Color Mapping	Assign color to nodes according to selected property of nodes.
Node Size Mapping	Assign the radius of nodes according to selected property of nodes.
Filter Node Label by Node Size	Selectively show the node label according to the threshold. Labels will be shown for the nodes of which the size is larger than the threshold.
Node Label Mapping	Set the label value according to selected property of nodes.
Node Label Size	Adjust the font size of node labels
Show Edges	Set the visibility of all edges
Edge Color Mapping	Assign color to edges according to selected property of edges.
Edge Label Mapping	Set the label value according to selected property of edges.
Edge Label Size	Adjust the font size of edge labels
Edge Thickness Mapping	Assign thickess to edges according to selected property of edges.
Edge Style	Select the rendering style of edges. For directed graphs, users also can choose if showing the arrows or not.

Visualization Panel – Before Customization

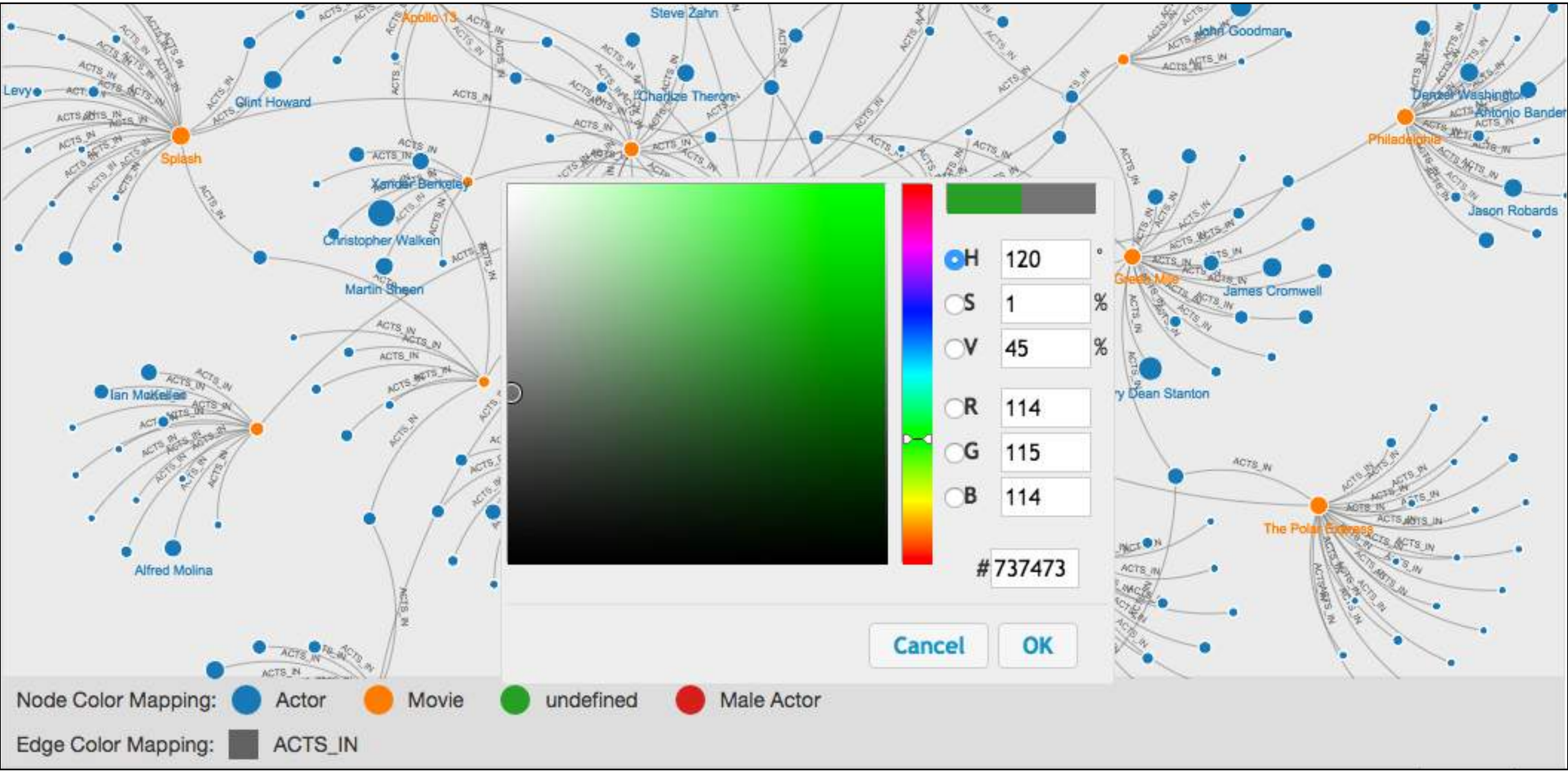


Visualization Panel – After Customization



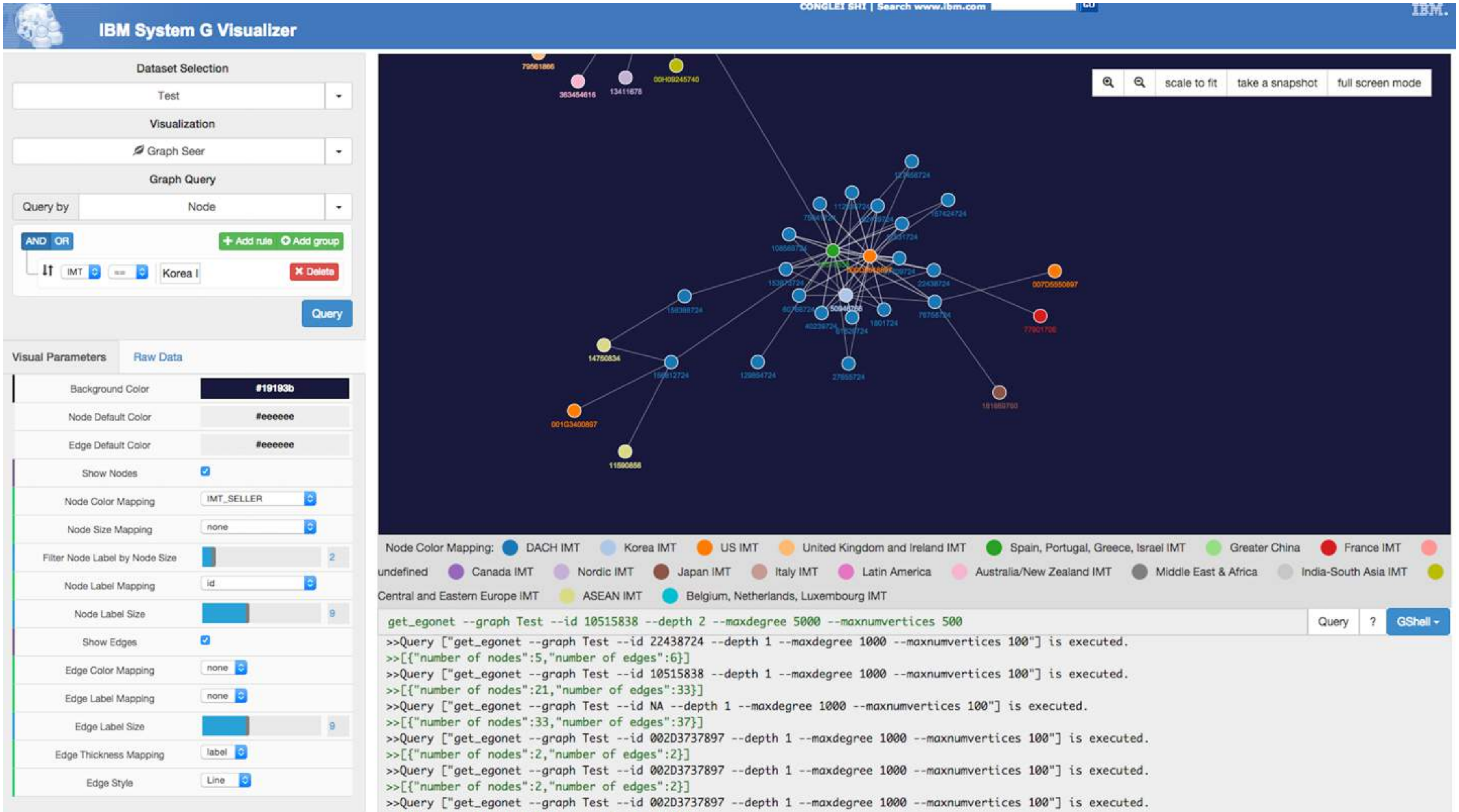
System G Team

Visualization Panel – Further Customization



Users can further specify colors by clicking the color blocks shown in the legend area

http://systemg.ibm.com/tool/visualizer/

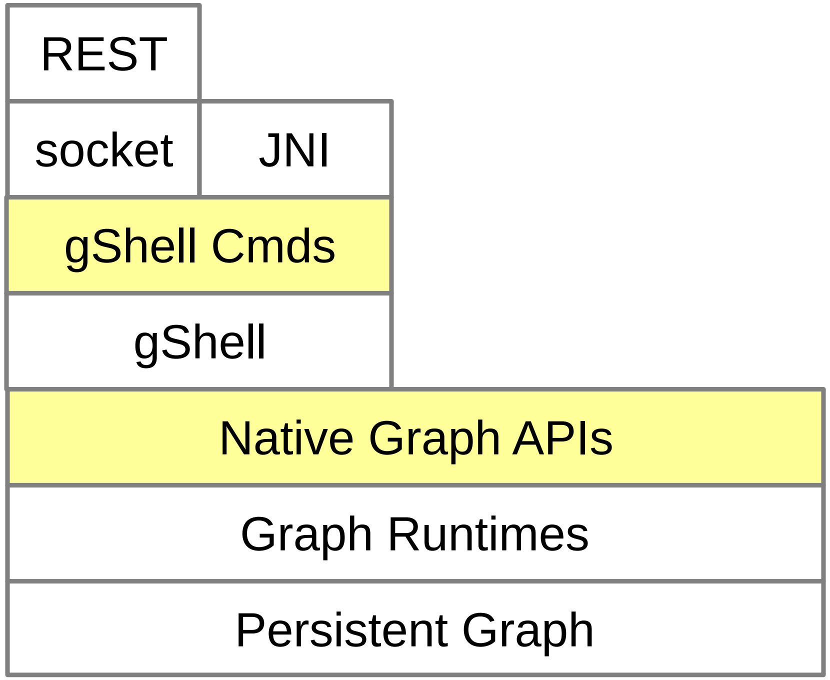


Quick Exploration of IBM System G

- gShell
- py-gShell
- gremlin-gShell
- REST API
- Programming/User-Defined Plugins

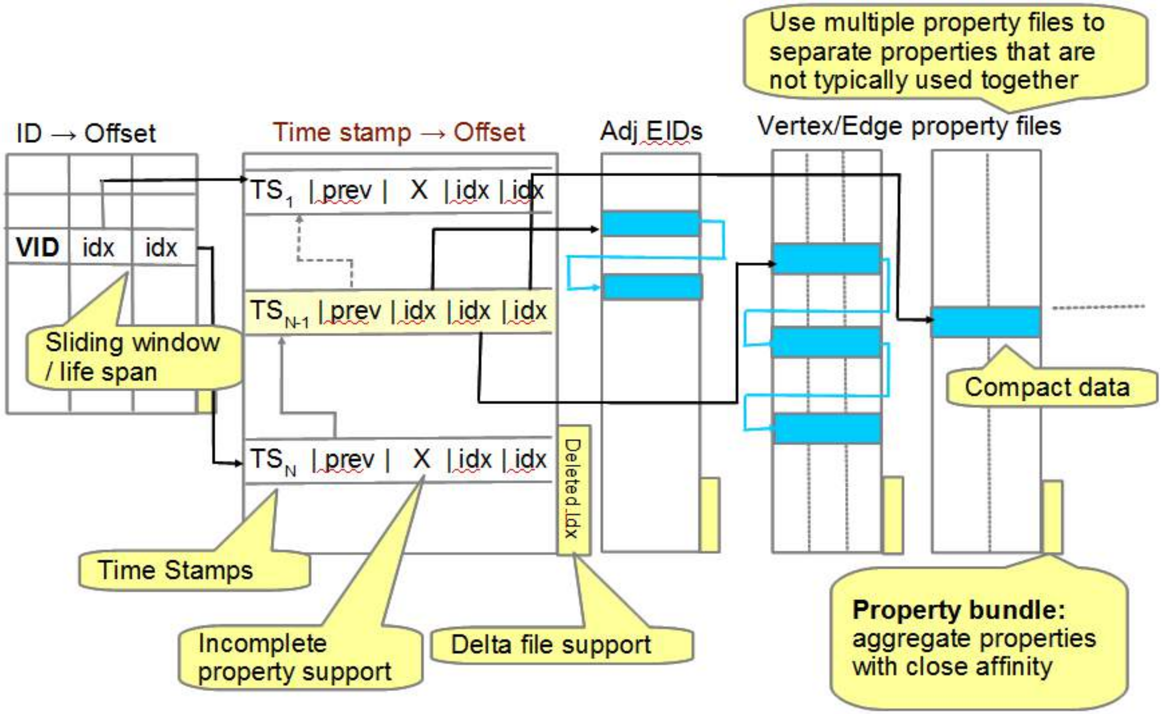
gShell → a Straightforward Way to Feel Native Store

- gShell is a simple implementation based on the native graph API
- Accepts a string of characters locally or remotely as the input
- Assumes properties are of identical format (can be k/v pairs)
- Outputs results with some format (plain text, json, etc) and interfaces with System G Visualization component
- Wrapped into C/S mode



gShell and IBM System G Native Store

- **Native store organizes graph data for representing a graph with both structure and the vertex properties and edge properties using multiple files in Linux file system**
  - Creating a list called ID → Offset where each element translates a vertex (edge) ID into two offsets, pointing to the earliest and latest data of the vertex/edge, respectively
  - Creating a list called Time_stamp → Offset where each element has a time stamp, an offset to the previous time stamp of the vertex/edge, and a set of indices to the adjacent edge list and properties
  - Create a list of chained block list to store adjacent list and properties



Download & Use — So Simple!

Download: http://systemg.research.ibm.com/download.html

Download and Support

The System G Graph Tools Trial Download version is **free**, intended for experimentation, research and application development. You can use it to support your commercial or non-commercial applications. But, please note that, this software cannot be redistributed or sold. It is the users' own risk using the software.

You can download the IBM System G Graph Tools Trial Download from here.

There is no online support for this version and IBM may choose to update the version at our discretion. Feedback & enhancement suggestions may be sent to systemg @ us . ibm . com (remove white space).

IBM System G > Download > Package

IBM System G Graph Tools Trial Download

Linux (CentOS 6.5 and Ubuntu 14.04)

IBM Power 8

Mac OS X

Use gShell

```
drwxr-xr-x  12 yxia  staff  408 Oct 27 09:52 systemg-tools-1.3.0_macosx-64bit
Yinglongs-MacBook-Pro:release yxia$ cd systemg-tools-1.3.0_macosx-64bit/
Yinglongs-MacBook-Pro:systemg-tools-1.3.0_macosx-64bit yxia$ ll
total 24
-rw-r--r--   1 yxia  staff  4109 Oct 14 13:10 README
drwxr-xr-x   9 yxia  staff   306 Oct 26 10:47 blueprints-gremlin
drwxr-xr-x  10 yxia  staff   340 Oct 14 11:56 data
drwxr-xr-x  11 yxia  staff   374 Oct 21 06:51 doc
drwxr-xr-x  12 yxia  staff   408 Oct 28 19:14 gshell
drwxr-xr-x  20 yxia  staff   680 Oct 14 13:02 lib
drwxr-xr-x  15 yxia  staff   510 Oct 28 10:20 python
drwxr-xr-x   9 yxia  staff   306 Oct 14 11:57 restapi
drwxr-xr-x  12 yxia  staff   408 Oct 27 09:52 systemg-lite
-rwxr-xr-x   1 yxia  staff   139 Oct 14 13:09 systemg.sh
Yinglongs-MacBook-Pro:systemg-tools-1.3.0_macosx-64bit yxia$
```

- `README`: a text file that describes the content of the package and provides references to documentation files
- `systemg.sh`: a script to set up environment variables required to run IBM System G Graph Tools
- `doc/`: documentation files
- `data/`: sample data files for tests
- `gshell/`: gShell executable files, sample data, and test scripts
- `lib/`: library files for gShell
- `python/`: Python interface to gShell
- `blueprints-gremlin/`: Blueprints API and Gremlin
- `resapi/`: REST API executable files and scripts
- `systemg-lite/`: IBM System G Lite visualization

Use gShell - 2

./gShell interactive

```
Yinglongs-MacBook-Pro:gshell yxia$ ./gShell  interactive
gShell>>
add_edge                      add_vertex                       add_vertex_json
analytic_auction              analytic_betweennes
analytic_closeness_centrality analytic_clustering
analytic_connected_component  analytic_degree_cen
analytic_k_core               analytic_k_shortest
analytic_reset_engine         analytic_shortest_p
analytic_stop_engine          analytic_triangle_c
delete_eprop                  delete_vertex
export_csv                    filter_edges
find_edge                     find_multiple_verti
find_random_edges             find_random_vertice
find_vertex_max_degree        get_egonet
get_num_vertices              get_subgraph
indexer_clucene               indexer_leveldb
load_csv_vertices             print_all
update_edge                   update_vertex
close                         close_all
delete                        create
help                          version
gShell>>
```

```
gShell>> list_all
[list_all]
{
"warning":[{"MESSAGE":"store is empty!"}]
}
gShell>> list_all --help
[list_all] [--help]
{
"info":[
{"MESSAGE":"list_all - list all graphs"},
{"MESSAGE":"--format:  [optional] output format"},
{"MESSAGE":"--help:  [optional] help infomation"}
]
}
```

See **help** here: http://systemg.ibm.com/doc/gshell.html

Write Python Code based on System G

```
#!/usr/bin/python
from py_gShell import _py_gshell as gShell
import json

g = gShell()
g.delete_graph("testu")
g.create_graph("testu", "undirected")
g.load_csv_vertices(csvfile="data/test.vertices.dat", keypos=0, labelpos=1)
g.load_csv_edges(csvfile="data/test.edges.dat", srcpos=0, targpos=1, labelpos=3)
g.add_vertex(vertex_id="7", label="C", prop={"tag":"T2","value":0.1})
g.add_vertex(vertex_id="8", prop={"value":0.4})
g.add_vertex(vertex_id="9", label="C", prop={"value":0.5})
g.update_vertex(vertex_id="9",prop={"value":0.55,"other":"1"})
g.add_edge(src="7", targ="8", edgelabel="c")
g.add_edge(src="7", targ="1", edgelabel="c",prop={"weight":8.0})
g.update_edge(src="1",targ="7", prop={"weight":8.6, "other":"2"})
g.add_edge(src="8",targ="9")
g.update_edge(src="1",targ="2", prop={"weight":6.5})
g.analytic_start_engine(edgeweightpropname="weight")
print json.dumps(json.loads(g.analytic_find_path(src="1",sink="2")), indent = 4)

print json.dumps(json.loads(g.analytic_find_path(src="1",sink="2",label="b")), indent = 4)

g.analytic_stop_engine()
```

Output of the above Python script

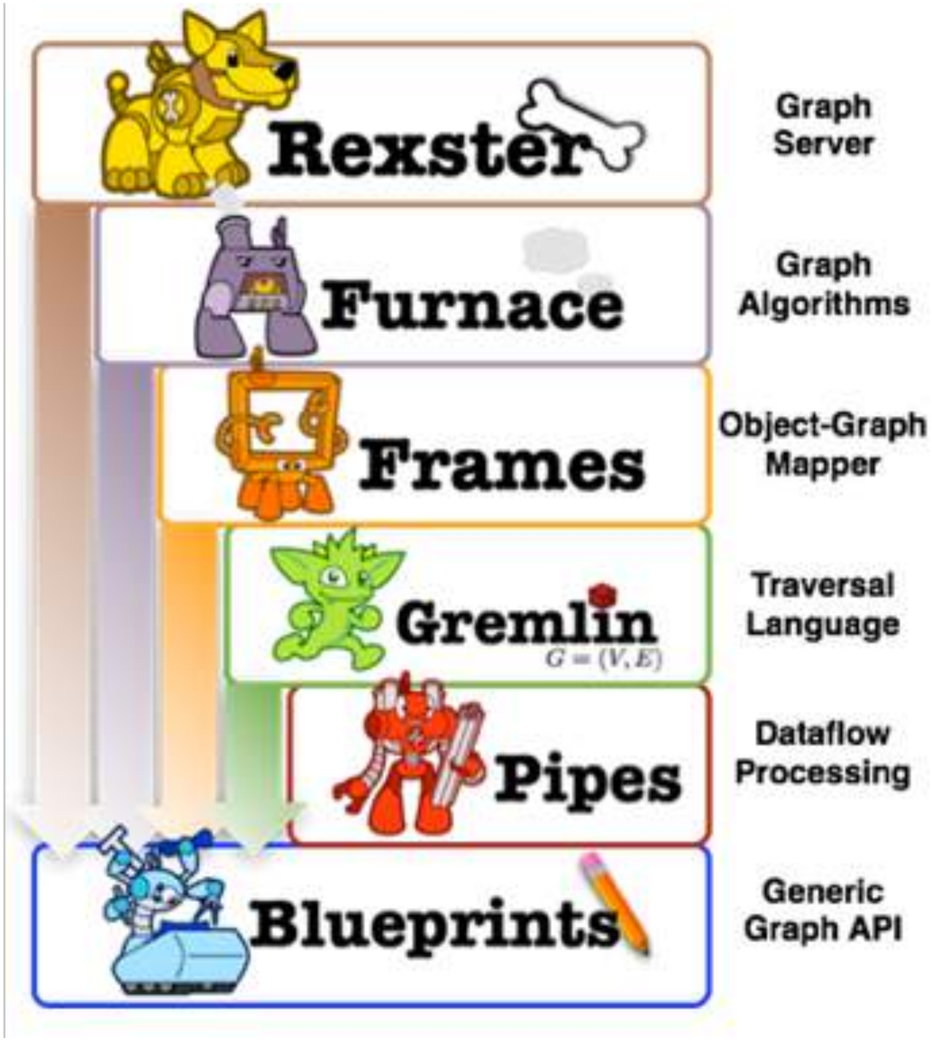
g.analytic_find_path(src="1",sink="2")


```
{
    "paths": [
        {
            "src": "1",
            "path": "1-->2",
            "sink": "2",
            "distance": 1.0
        }
    ],
    "time": [
        {
            "TIME": "3.31402e-05"
        }
    ]
}
```


g.analytic_find_path(src="1",sink="2",label="b")


```
{
    "paths": [
        {
            "src": "1",
            "path": "1-->3-->5-->2",
            "sink": "2",
            "distance": 3.0
        }
    ],
    "time": [
        {
            "TIME": "2.09808e-05"
        }
    ]
}
```

Open Source TinkerPop Stack (Apache Incubator)



http://sql2gremlin.com

http://tinkerpop.incubator.apache.org

Use Gremlin-gShell

```
gremlin> g = CreateGraph.openGraph("nativemem_authors","awesome")
==>nsgraph[vertices:7 edges:8]
gremlin> g.class
==>class  com.ibm.research.systemg.nativestore.tinkerpop.NSGraph
gremlin> // lets look at all the vertices
==>true
gremlin> g.V
```

```
gremlin> gs = new GShell()
==>com.ibm.research.systemg.nativestore.gshell.GShell@5e88a3de
gremlin> gs.exec("create --graph test --type directed")
140711320353584
[create] [--graph] [test] [--type] [directed]
==>{
"info":[{"MESSAGE":"store [test] is created!"}]
}
gremlin> gs.exec("add_vertex --graph test --id \"test node\" --prop tag:\"test tag\"")
139868232521952
[add_vertex] [--graph] [test] [--id] [test node] [--prop] [tag] [test tag]
==>{
"info":[{"MESSAGE":"vertex is added"}],
"time":[{"TIME":"0.000422001"}]
}
```

User-Defined Analytics

- a header file template
- a cpp file template
- add .o file to link
  THAT'S IT!

```
#ifndef _PLUGIN_HELLOWORLD_H
#define _PLUGIN_HELLOWORLD_H

#include "defines.hpp"
#include "types.hpp"
#include "string_parser.hpp"
#include "query_map.h"

class example_helloWorld : public query_base
{
 public:
  REGISTER_QUERY_TYPE(example_helloWorld);

  void options(command_options &opt);
  int  run(struct query_param_type param);
};

#endif
```

```
#include "plugin_helloWorld.h"

REGISTER_QUERY_NAME(example_helloWorld, "example_helloWorld");

void example_helloWorld::options(command_options &opts)
{
  opts.add_command_info("this is an example of gShell plugin");
  opts.add_option("arg1", true, HAS_ARGUMENT, "arg1 is a mandatory argument with valu
  opts.add_option("arg2", false, HAS_ARGUMENT, "arg2 is an optional argument with val
  opts.add_option("arg3", false, NO_ARGUMENT, "arg3 is an optional flag");
  opts.add_option("arg4", false, MULTIPLE_ARGUMENT, "arg4 is an optional flag with mu
}

int example_helloWorld::run(struct query_param_type param)
{
  if (param.directness == TYPE_UNDIRECT)
    param.internal_output->info("this is a undirected graph");
  else
    param.internal_output->warning("this is a directed graph");

  string arg1, arg2;
  param.opts->get_value("arg1", arg1);
  param.opts->get_value("arg2", arg2);

  bool arg3 = param.opts->get_flag("arg3");
  if (arg3)
    param.internal_output->info("arg3 is true");
  else
    param.internal_output->info("arg3 is false");
```

IBM System G Eco-System (GraphBIG)

GraphBIG

A group of graph analytics for benchmarking underlying platforms

A simplified IBM System G in-memory graph layer, with similar APIs

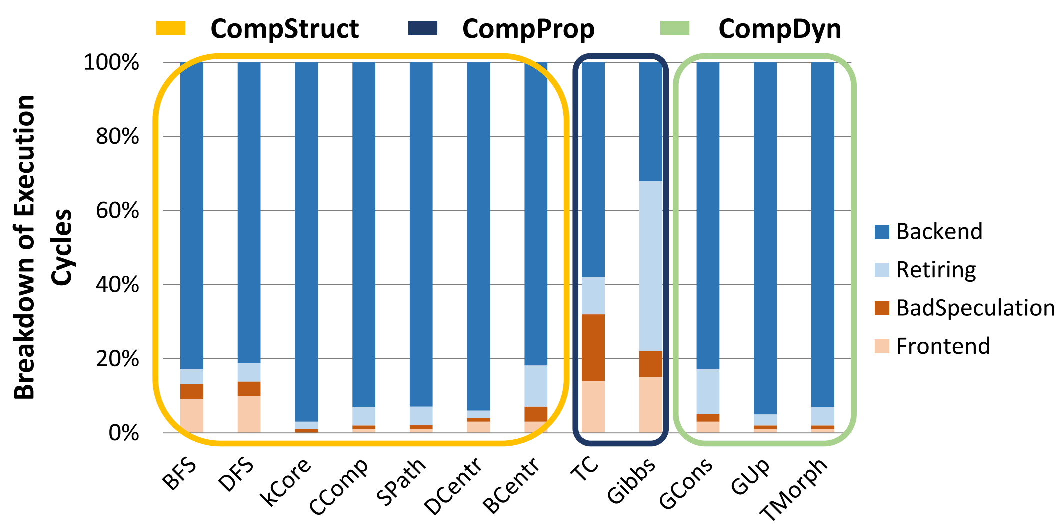
Come with performance profiler by taking hardware performance counters, breaking down the execution time into multiple stages to reveal the performance bottleneck

Fetch Code

Code: https://github.com/graphbig/graphBIG

Doc: https://github.com/graphbig/GraphBIG-Doc

```
-bash:~$ git clone https://github.com/graphbig/graphBIG.git GraphBIG
Cloning into 'GraphBIG'...
remote: Counting objects: 497, done.
remote: Compressing objects: 100% (110/110), done.
remote: Total 497 (delta 57), reused 0 (delta 0), pack-reused 386
Receiving objects: 100% (497/497), 2.07 MiB | 0 bytes/s, done.
Resolving deltas: 100% (229/229), done.
Checking connectivity... done.
-bash:~$
```

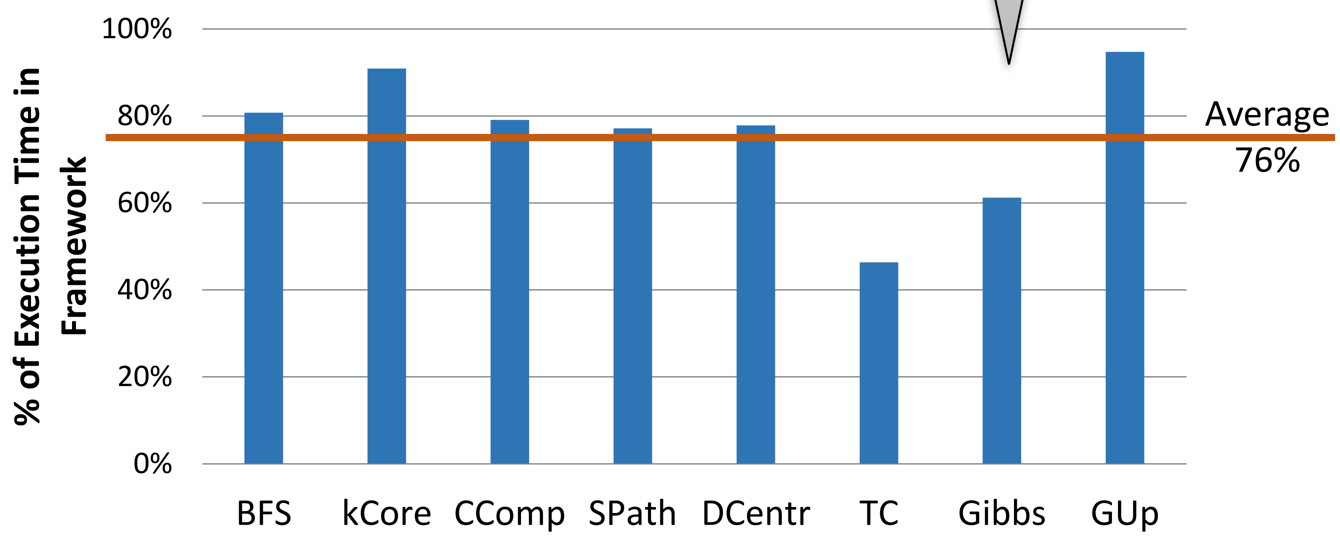


System G Team

Understand Graph Computational Challenges

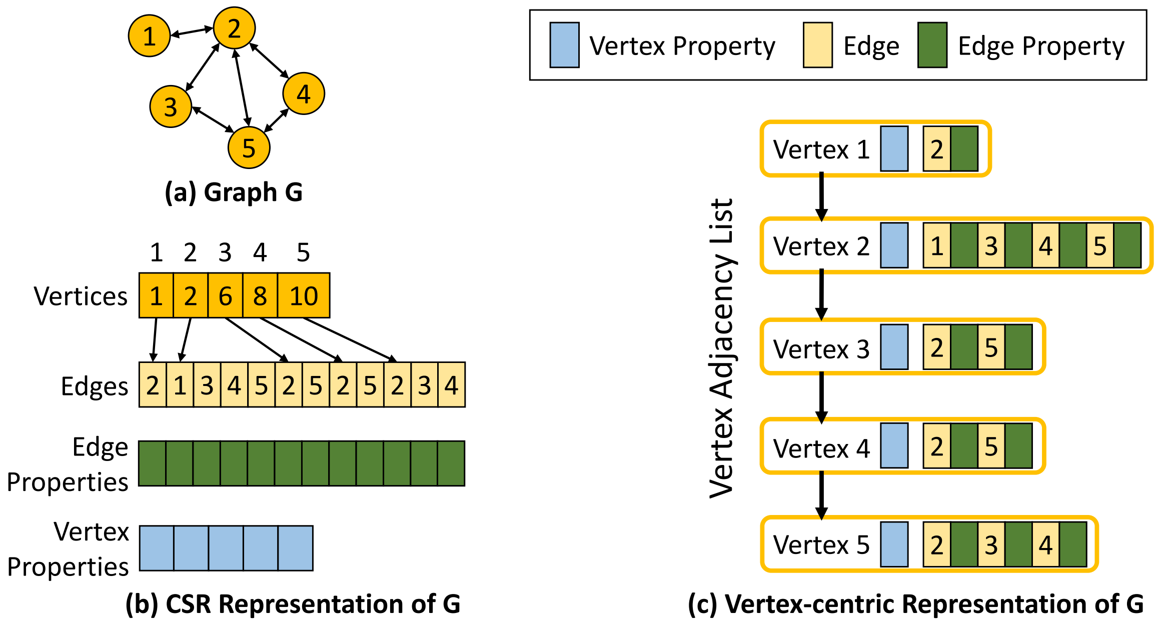
76%

76% of the total execution is spent inside the framework by invoking primitive graph operations framework



framework actually plays a critical role

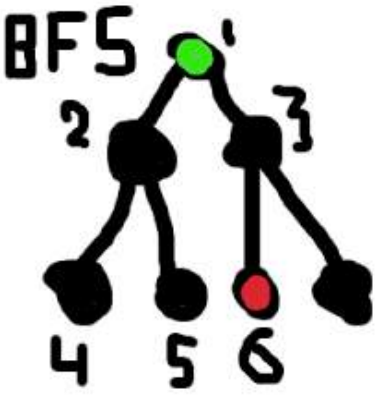
Graph Data Representations



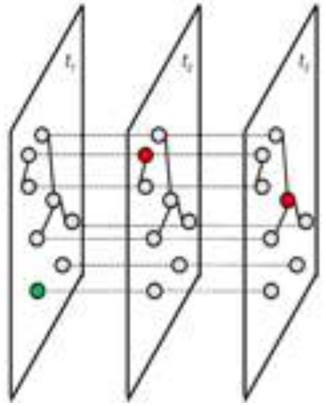
CSR format is compact, and maybe good for cache performance. But it is static, and cannot support structure changes. However, in practices, graphs are usually dynamic. This is why vertex-centric representation is popular across multiple graph frameworks.

Graph Computing Types

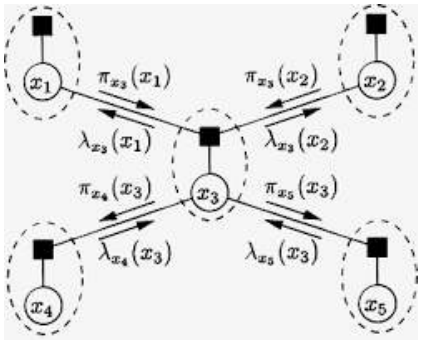
- Computation on graph structure (CompStruct)
  - Example: Breadth-first search
  - Irregular access pattern, heavy read access



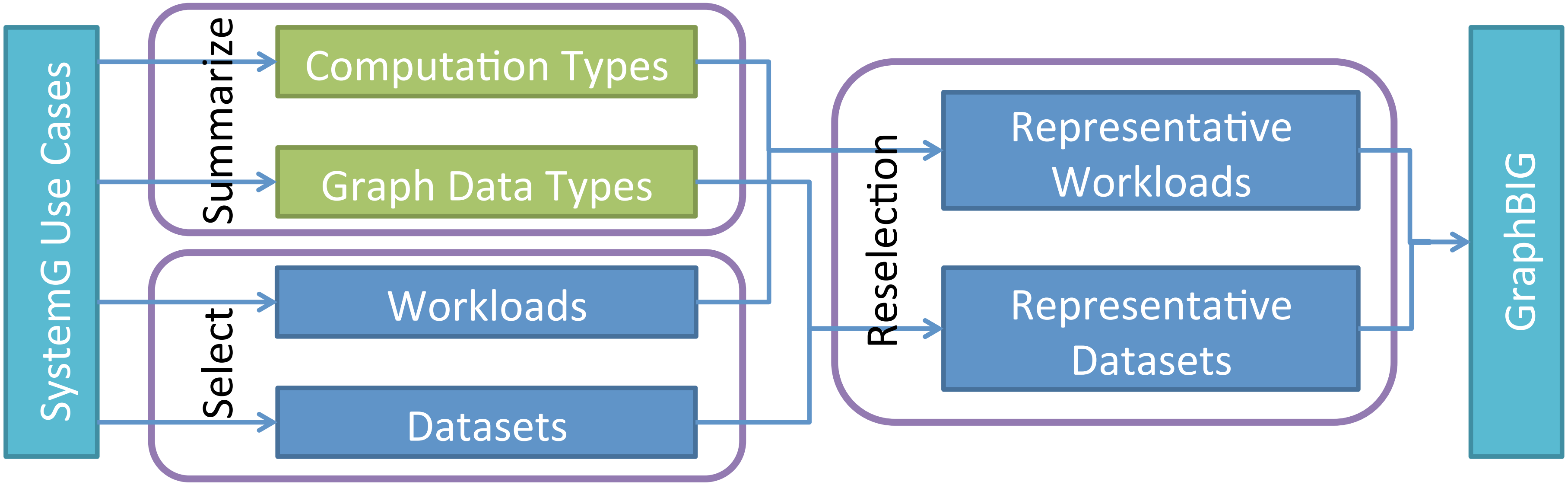
- Computation on dynamic graph (CompDyn)
  - Example: Streaming Graph
  - Dynamic graph structure, dynamic memory usage



- Computation on graph property (CompProp)
  - Example: Belief propagation
  - Heavy numeric operations on graph property

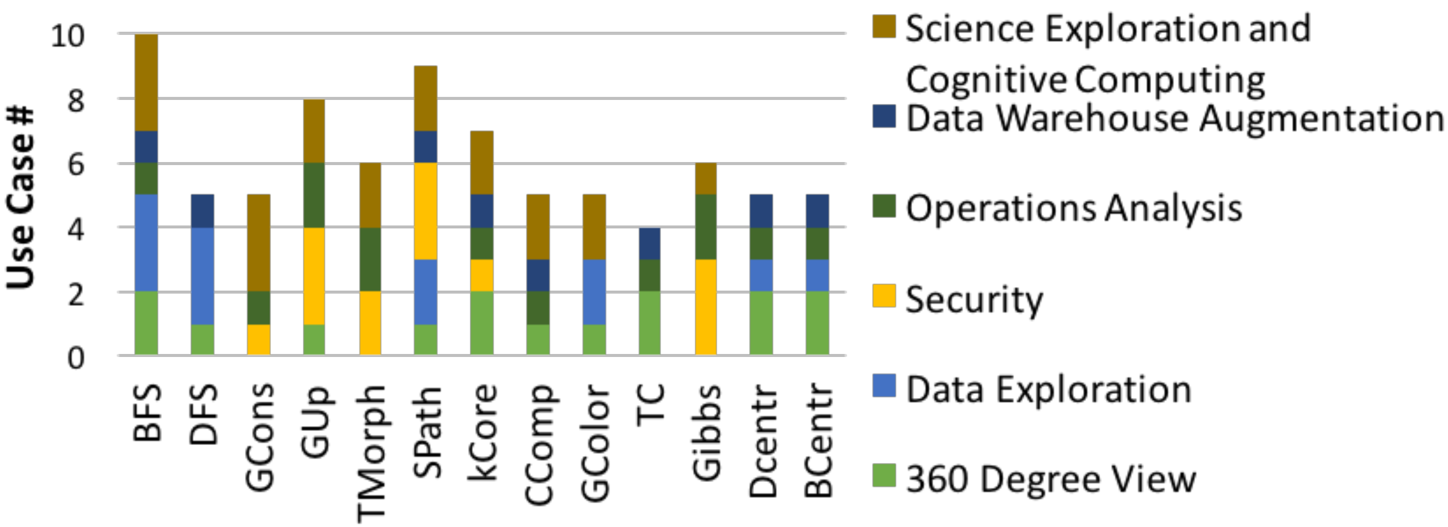


Graph Workload Selection to Form a Benchmark



We start from the use cases of IBM System G. By analyzing the use cases, we are able to summarize the computation and data types. Meanwhile, we select workloads and data from them. After that, we then have a reselection stage. In the reselection stage, we reselect workloads and data to ensure that they cover all computation and data types.

Workload Selection



(A) # of Use Cases with Each Workload

(B) Distribution of Selected Use Cases in 6 Categories

In total, we analyzed 21 use cases from 6 different categories, from science exploration to security.

Different categories contain different use cases and different selected workloads also have different popularities across the use cases. But in general, all workloads are widely used in multiple real-world use cases.

Workload Summary and Experiments to Show

Category	Workload	Computation Type	CPU	GPU
Graph traversal	BFS	CompStruct	✔	✔
	DFS	CompStruct	✔	
Graph update	Graph construction (GCons)	CompDyn	✔	
	Graph update (GUp)	CompDyn	✔	
	Topology morphing (TMorph)	CompDyn	✔	
Graph analytics	Shortest path (SPath)	CompStruct	✔	✔
	kCore	CompStruct	✔	✔
	Connected component (CComp)	CompStruct	✔	✔
	Graph coloring (GColor)	CompStruct		✔
	Triangle counting (TC)	CompProp	✔	✔
	Gibbs Inference (GI)	CompProp	✔	
Social analytics	Betweenness Centrality (BCentr)	CompStruct	✔	✔
	Degree Centrality (DCentr)	CompStruct	✔	✔

Data set	Type	Vertex #	Edge #
Twitter Graph	Type 1	120M	1.9B
IBM Knowledge Repo	Type 2	154K	1.72M
IBM Watson Gene Graph	Type 3	2M	12.2M
CA Road Network	Type 4	1.9M	2.8M
LDBC Graph	Synthetic	Any	Any

GraphBIG Hands-on

GraphBIG is open sourced under BSD license. We have an organization in github named as graphbig.
To obtain the GraphBIG code is pretty simple. Just do use git to perform a "git clone"
More detailed documents can also be found in a separate repository in the same organization in github.

Fetch Code

Code: https://github.com/graphbig/graphBIG
Doc: https://github.com/graphbig/GraphBIG-Doc

```
-bash:~$ git clone https://github.com/graphbig/graphBIG.git GraphBIG
Cloning into 'GraphBIG'...
remote: Counting objects: 497, done.
remote: Compressing objects: 100% (110/110), done.
remote: Total 497 (delta 57), reused 0 (delta 0), pack-reused 386
Receiving objects: 100% (497/497), 2.07 MiB | 0 bytes/s, done.
Resolving deltas: 100% (229/229), done.
Checking connectivity... done.
-bash:~$
```

GraphBIG Hands-on - 2

GraphBIG is a standalone package. It doesn't require any external libraries. But of course, you need a gcc and for gpu workloads, you need cuda sdk
To compile it, just "make all".
To compile the full suite, you can "make all" at the top level. If you just want to compile CPU benchmarks, get into "benchmark/" directory and "make all". Similarly for GPU workloads, get into "gpu_bench/" and "make all"

Compile

- Require: gcc/g++ (>4.3), gnu make
- Just *"make all"*

```
-bash:~$ cd GraphBIG/
-bash:GraphBIG$ ls
benchmark  CHANGELOG.md  common  csr_bench  dataset  gpu_bench  LICENSE  openG  README.md  tools
-bash:GraphBIG$ cd benchmark/
-bash:benchmark$ ls
bench_betweennessCentr  bench_degreeCentr      bench_graphConstruct  bench_shortestPath    common.mk   ubench_add     ubench_traverse
bench_BFS               bench_DFS              bench_graphUpdate     bench_TopoMorph       Makefile    ubench_delete
bench_connectedComp     bench_gibbsInference   bench_kCore           bench_triangleCount   README.txt  ubench_find
-bash:benchmark$ make all
make -C ../tools all
make[1]: Entering directory `/home/lifeng/GraphBIG/tools'
rm -rf libpfm-4.5.0
tar xzvf libpfm-4.5.0.tar.gz
libpfm-4.5.0/./
libpfm-4.5.0/./COPYING
libpfm-4.5.0/./lib/
libpfm-4.5.0/./lib/pfmlib_ppc970.c
libpfm-4.5.0/./lib/pfmlib_powerpc.c
libpfm-4.5.0/./lib/pfmlib_sparc_priv.h
libpfm-4.5.0/./lib/pfmlib_powerpc_perf_event.c
```

GraphBIG Hands-on - 3

Test Run

Just *"make run"*

Using default "small" dataset

Help info: ./<exe> --help

```
-bash:benchmark$ cd bench_BFS/
-bash:bench_BFS$ make run
Running bfs, output in output.log
```

```
-bash:bench_BFS$ cat output.log
==================================================================
   ________                    .__     __________.___  ________
  /  _____/___________  ______ |  |__ \______   \   |/  _____/
 /   \  ___\_  __ \__  \ \____ \|  |  \ |    |  _/   /   \  ___
 \    \_\  \|  | \// __ \|  |_> >   Y  \|    |   \   \    \_\  \
  \______  /|__|  (____  /   __/|___|  /______  /___|\______  /
         \/             \/|__|        \/       \/            \/

==================================================================
Benchmark: BFS
loading data...
== 1000 vertices  29790 edges
== time: 0.0530188 sec

BFS root: 31
BFS finish:
== time: 0.00118506 sec
PERF_COUNT_HW_CPU_CYCLES          ==>     2581036
PERF_COUNT_HW_INSTRUCTIONS        ==>     774222
PERF_COUNT_HW_BRANCH_INSTRUCTIONS         ==>     200529
PERF_COUNT_HW_BRANCH_MISSES       ==>     10486
PERF_COUNT_HW_CACHE_L1D_READ_ACCESS       ==>     309284
PERF_COUNT_HW_CACHE_L1D_READ_MISS         ==>     99740
==================================================================
```

It is also pretty simple to make a test run of GraphBIG workloads. We include the simple test run already in the makefile. You can get into the directory of any benchmark and use "make run". Then, a test run will be performed and the output will be stored in a log file named "output.log"

To get more info about the arguments of a specific benchmark, just run it with "--help"

Characterization

Methodology

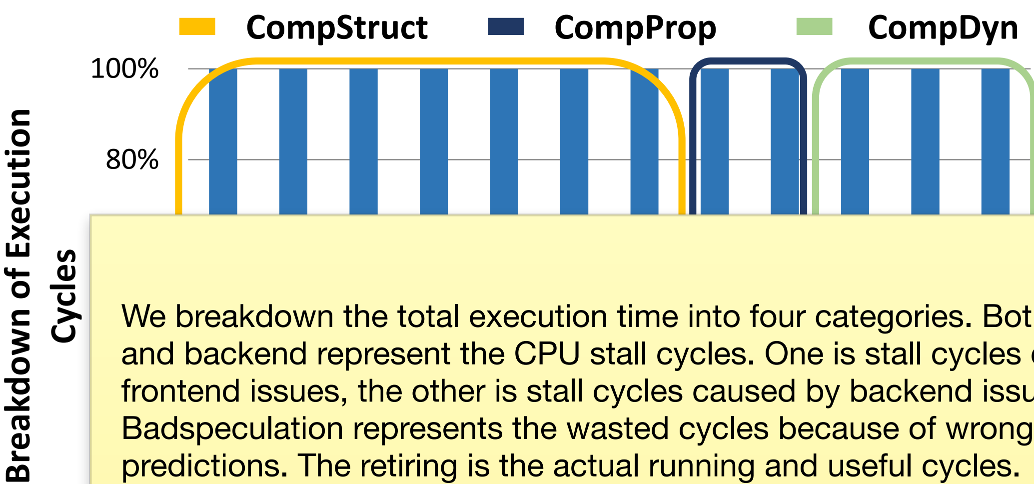
Real machine + hardware performance counters

CPU: linux perf event kernel calls (integrated with benchmarks)

GPU: CUDA nvprof

Processor	**Type**	**Xeon E5-2670**
	Frequency	2.6 GHz
	Core #	2 sockets x 8 cores x 2 threads
	Cache	32KB L1, 256KB L2, 20MB L3
	Memory BW	51.2 GB/s (DDR3)
GPU	Type	Nvidia Tesla K40
	CUDA Core	2880
	Memory	12 GB
	Memory BW	288 GB/s
	Frequency	Core-745 MHz, mem-3 GHz
System	Memory	192 GB
	Disk	2 TB HDD
	OS	RHEL 6

Execution Time Breakdown



Backend is the bottleneck

We breakdown the total execution time into four categories. Both frontend and backend represent the CPU stall cycles. One is stall cycles caused by frontend issues, the other is stall cycles caused by backend issues. Badspeculation represents the wasted cycles because of wrong branch predictions. The retiring is the actual running and useful cycles.

We can see that for most workloads, backend is the dominant, it is the bottleneck here. Backend may include instruction execution, retiring, memory sub-systems.
But outliers also exsit, for TC (triangle counting) and Gibbs (gibbs inference), they are not suffering from backend issues.
It shows an interesting diversity across benchmarks

ation

he total cycles being spent by the CPU in 3 categories:

re instructions get retired (usefull work)

g spent in the Back-End (wasted)

nt in the Front-End (wasted).

Th … rces
(us … tc.).

The **cycles stalled in the front-end** are a waste because that means that the CPU does not feed the Back End with instructions. This can mean that you have misses in the Instruction cache, or complex instructions that are not already decoded in the micro-op cache.

IBM System G Eco-System (ScaleGraph)

IBM Research

TOKYO TECH Pursuing Excellence

UCD DUBLIN

JST Japan Science and Technology Agency

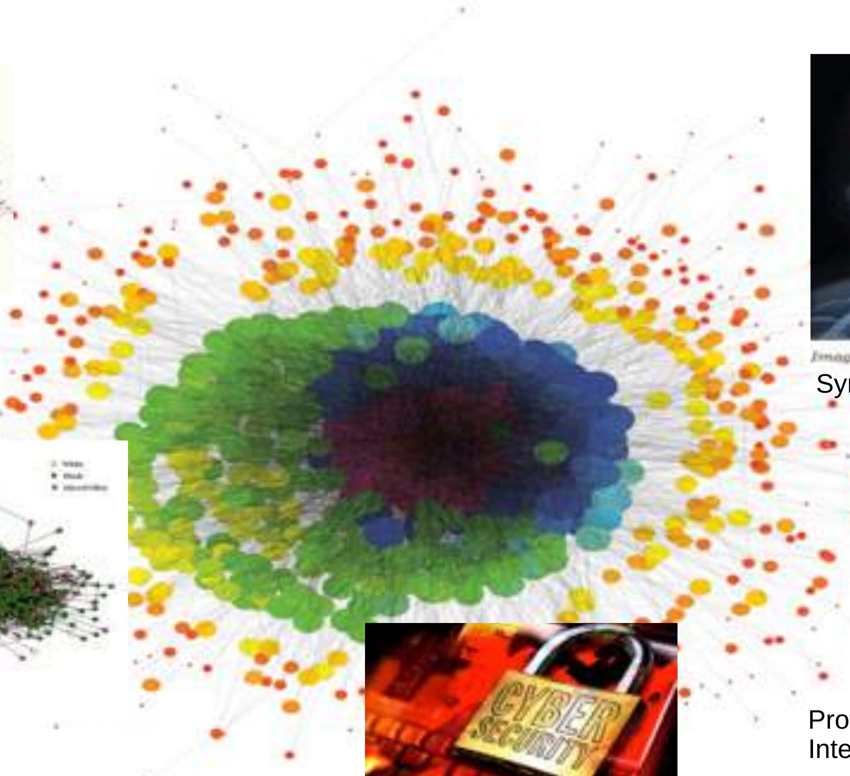
ScaleGraph Library

Build an open source **Highly Scalable Large Scale Graph Analytics Library** beyond the scale of billions of vertices and edges on Distributed Systems

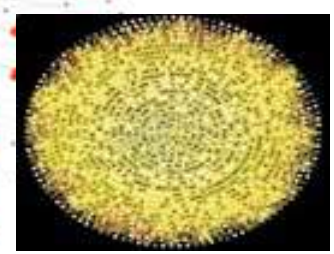
Internet Map

Image: Illustration by Mirko Ilic

Symbolic Networks:



Social Networks



Protein Interactions

Cyber Security (15 billion log entries / day for large enterprise)

Graph Algorithms

Currently supported algorithms

- PageRank
- Degree Distribution
- Betweenness Centrality
- Shortest path
- Breadth First Search
- Minimum spanning tree (forest)
- Strongly connected component
- Spectral clustering
- Separation of Degree (HyperANF)
- Cluster Coefficient

The algorithms that will be supported in the future.

- Blondel clustering
- Eigen solver for sparse matrix
- Connected component
- Random walk with restart
- etc.

Weak Scaling and Strong Scaling Performance up to 128 nodes (1536 cores)

Weak Scaling Performance of Each Algorithm (seconds): RMAT Graph of Scale 22 per node

	PageRank	BFS	SSSP	WCC	SC	HyperANF	Degree
RMAT, Scale 22, 1 nodes	13.7	1.9	8.9	5.6	351.1	50.3	33.1
RMAT, Scale 26, 16 nodes	28.3	4.0	13.5	12.0	701.4	88.9	36.3
RMAT, Scale 28, 64 nodes	37.9	7.5	18.8	17.0	1166.0	103.5	39.4
RMAT, Scale 29, 128 nodes	45.3	11.2	24.5	22.1	1438.8	142.3	41.1
Random, Scale 29, 128 nodes	46.5	8.8	20.6	21.4	1106.6	162.3	42.7

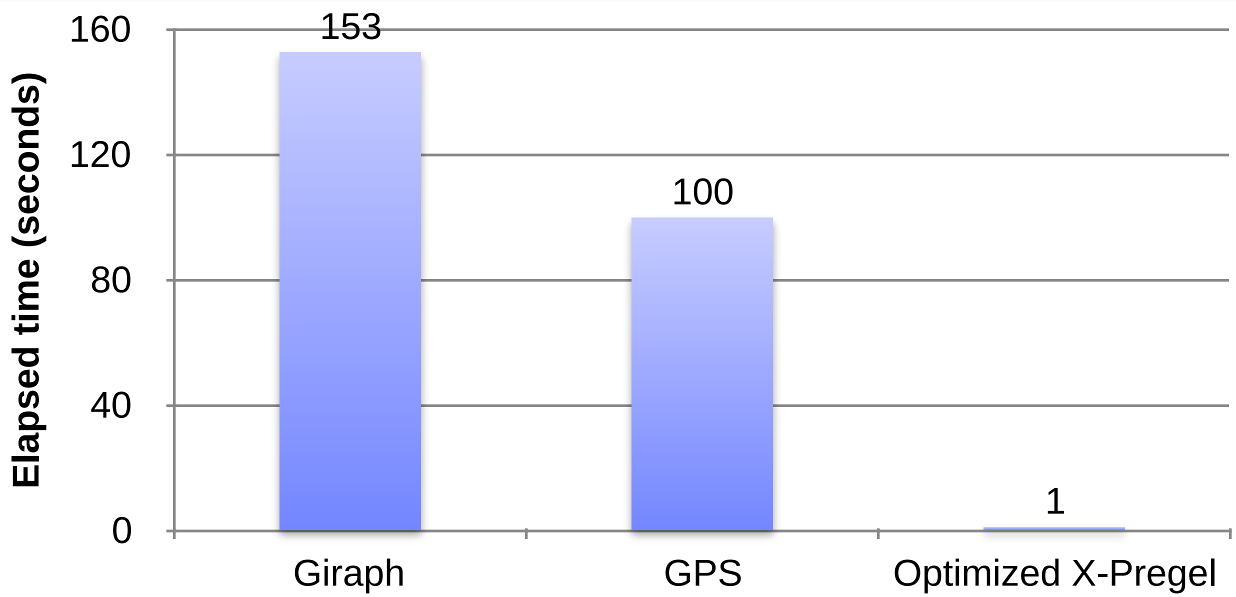
Strong Scaling Performance of Each Algorithm (seconds): RMAT Graph of Scale 28

	PageRank	BFS	SSSP	WCC	SC	HyperANF	Degree
16 nodes	124.1	21.9	65.8	55.9	2969.9	38.0	16.1
32 nodes	91.7	18.7	36.9	30.2	1639.0	27.0	11.6
64 nodes	38.1	7.5	20.1	17.2	1169.9	10.6	4.9
128 nodes	26.5	5.8	14.7	10.5	706.4	6.8	3.1

Evaluation Environment: TSUBAME 2.5 (Each node is equipped with two Intel® Xeon® X5760 2.93 GHz CPUs by each CPU having 6 cores and 12 hardware threads, 54GB of memory. All compute nodes are connected with InifinitBand QDR

Performance of XPregel

The execution time of PageRank 30 iteration for the Scale 20 (1million vertices, 16 million edges) RMAT graph with 4 TSUBAME nodes.



Framework	Execution Time (second)
Giraph	153
GPS	100
Optimized X-Pregel	2.4

Giraph and GPS data is from [Bao and Suzumura, LSNA 2013 WWW Workshop].

Web Site and Source Code Repository

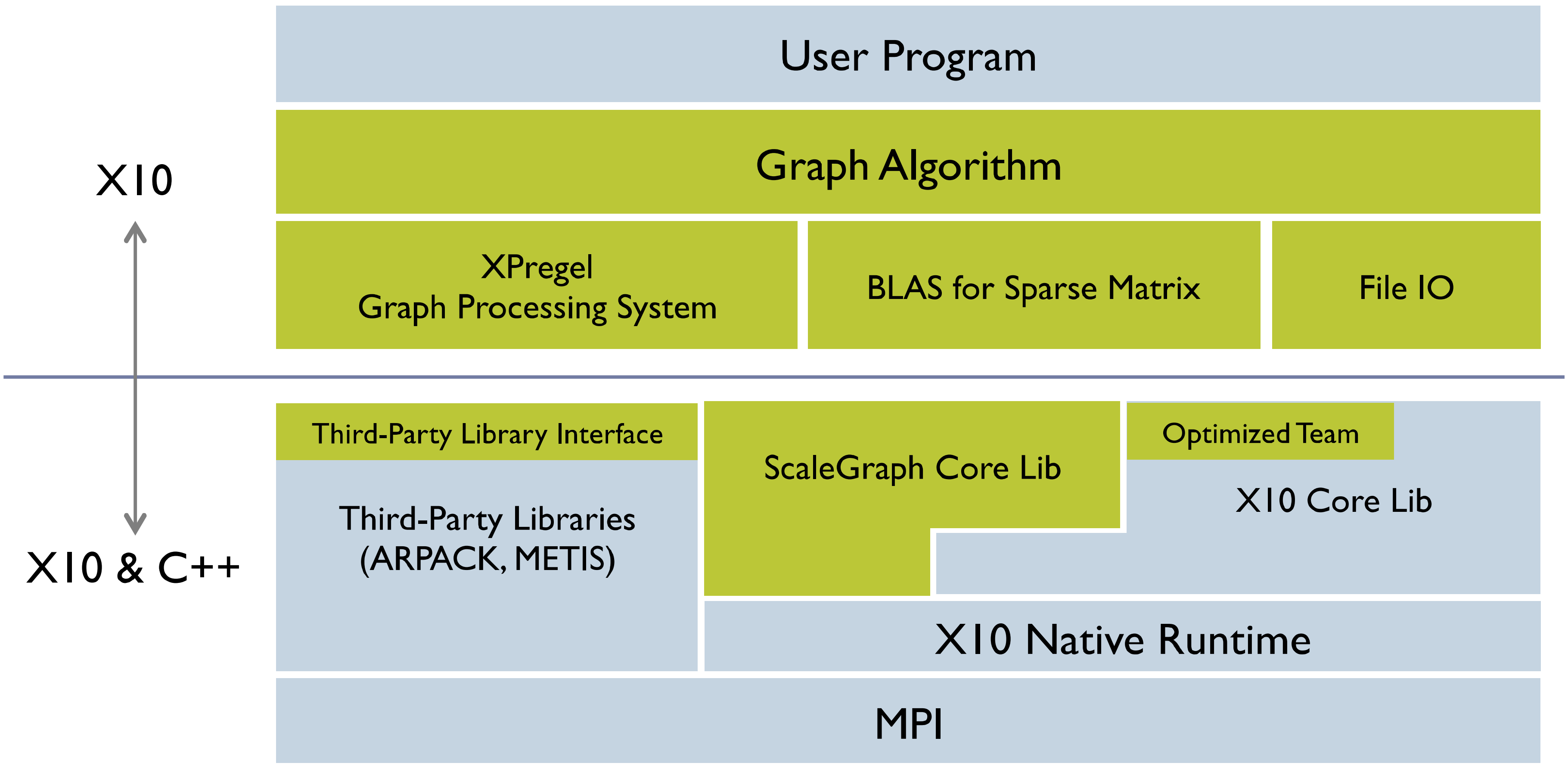
Official web site – http://scalegraph.org

- Project information
- Source code distribution
- Documentation

Source code repository - http://github.com/scalegraph/

- License: Eclipse Public License v1.0
- Project information and Documentation
- Source code distribution / VM Image

ScaleGraph Software Stack



Developing Graph Algorithms (e.g. PageRank)

```
xpgraph.iterate[Double,Double](
// Compute closure
(ctx :VertexContext[Double, Double, Double, Double], messages :MemoryChunk[Double]) => {
        val value :Double;
        if(ctx.superstep() == 0) {
          // calculate initial page rank score of each vertex
            value = 1.0 / ctx.numberOfVertices();}
        else {
        // for step onward,
          value = (1.0-damping) / ctx.numberOfVertices() +
              damping * MathAppend.sum(messages);}
        // sum score
        ctx.aggregate(Math.abs(value - ctx.value()));
        // set new rank score
        ctx.setValue(value);
        // broadcast its score to its neighbors
        ctx.sendMessageToAllNeighbors(value / ctx.outEdgesId().size());
},

// Aggregate closure: calculate aggregate value
(values :MemoryChunk[Double]) => MathAppend.sum(values),
// End closure : should continue ?
(superstep :Int, aggVal :Double) => {
        return (superstep >= maxIter || aggVal < eps);
});
```

```
public def iterate[M,A](
    compute :(ctx:VertexContext [V,E,M,A],
                messages:MemoryChunk[M])
    => void,
    aggregator :(MemoryChunk[A])=>A,
    end :(Int,A)=>Boolean)
```

Developing Graph Algorithms (e.g. PageRank)

- The core algorithm of a graph kernel can be implemented by calling ***iterate*** method of XPregelGraph as shown in the example.
- Users are also required to specify the type of messages (M) as well as the type of aggregated value (V).
- The method accepts three closures: *compute* closure, *aggregator* closure, and *end* closure.
- In each superstep (iteration step), a vertex contributes its value, which depends on the number of links, to its neighbors.
- Each vertex summarizes the score from its neighbors and then set the score as its value.
- The computation continues until the aggregated value of change in vertex's value less than a given criteria or the number of iterations less than a given value.

Installation and Developing Graph Algorithms

Installation and Execution Guide

http://www.scalegraph.org/web/index.php/documentation/getting-started-guides

PageRank Example:

https://github.com/scalegraph/scalegraph/blob/develop/src/example/PageRankSimple.x10

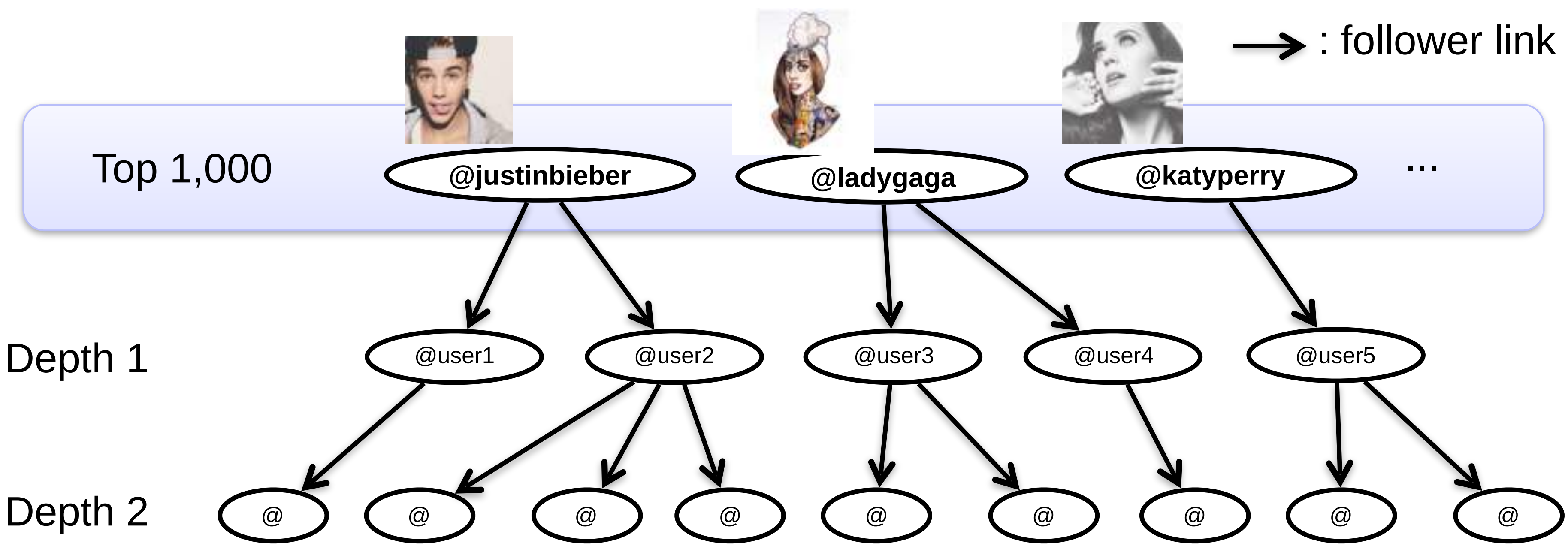
Understanding time-series nature of large-scale social networks (e.g. separation of degree, diameter, clustering, ..)

Crawled the entire Twitter follower/followee network of **826.10 million vertices and 29.23 billion edges. How could we analyze this gigantic graph ?**

Supercomputers

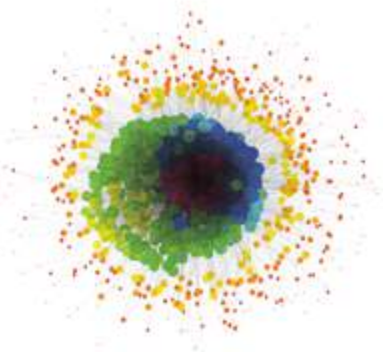
Crawling Billion-Scale Twitter Follower-Followee Network

- with Twitter API (v1.0*) from Jul. 2012 to Oct. 2012 (around 3 months).
- begin with top 1,000 users[*1] with the largest number of followers
- according to breadth-first search along the direction of follower



*1 : Twitaholic. http://twitaholic.com/top100/followers/

Crawled Data Set



- We stopped our crawling at depth 29
  - Because the user after depth 26 was less than 100.
  - Finally, we collected **469.9 million user data**.
- Collect two kind of user data by crawling for 3 months
  - 1. User profile
    - Include user id, screen_name, description, account creation time, time zone, etc.
    - The serialized data size is **91GB**
  - 2. Follower-friend
    - Adjacency list of followers and friends
    - The compressed(gzip) data size is **231GB**
- To perform the Twitter network analysis
  - **Apache Hadoop** for large-scale data processing
  - **HyperANF** for approximate calculation of degree of separation and diameter
    - Lars Backstrom[*1] also use HyperANF for Facebook network analysis

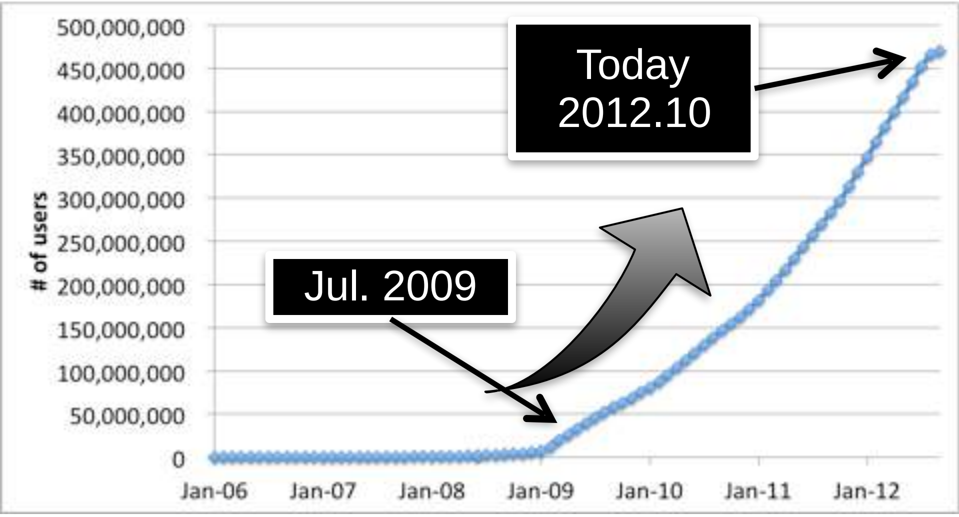
*1 : "Four degrees of separation" *ACM Web Science 2012*

Explore Twitter Evolution (1/2)
- Transition of the number of users

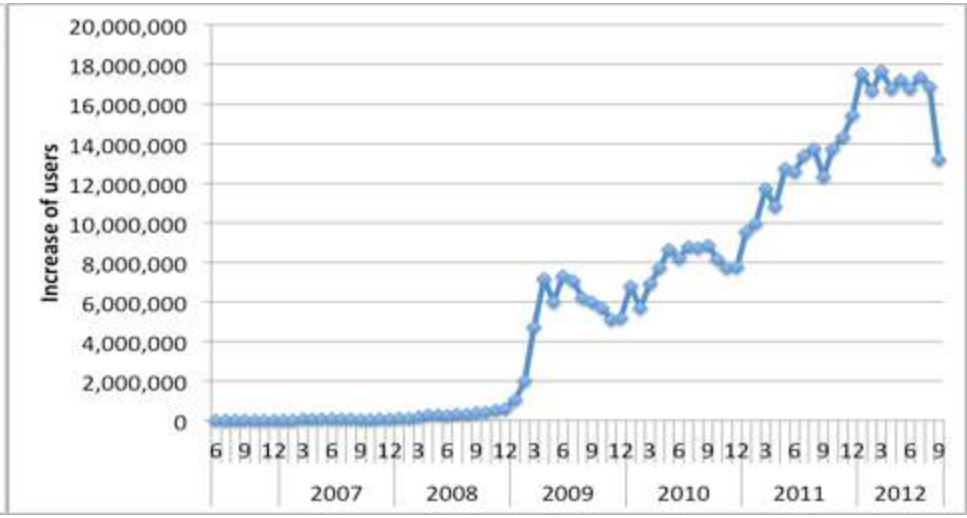
Total user count (left fig.)
- Twitter started at June 2006 and rapidly expanded from beginning of 2009.
- Haewoon Kwak *1 studied Twitter network on July 2009

Monthly increase of users (right fig.)
- Twitter users increase, but it seems a little unstable...

Total user count



Monthly increase of users



*1 : "What is Twitter, a social network or a news media?"

Explore Twitter Evolution (2/2)
- Transition of the number of users by regions-

Classify 131 million users by “Time zone” property under 6 regions

Africa, Asia, Europe, Latin America and Caribbean (Latin), Northern America (NA), Oceania

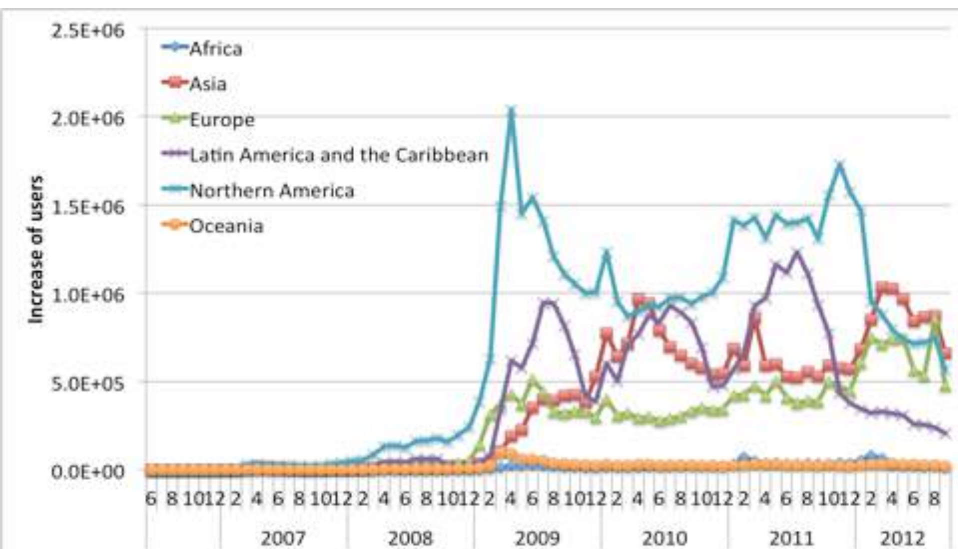
<u>Only 131 million user correctly set one’s own “Time zone”</u>

Massive change of ratio of users by region

Asia users : 8.30% => 20.8% (12.5% up)

NA users : 54.4% => 40.4% (14.0% down)

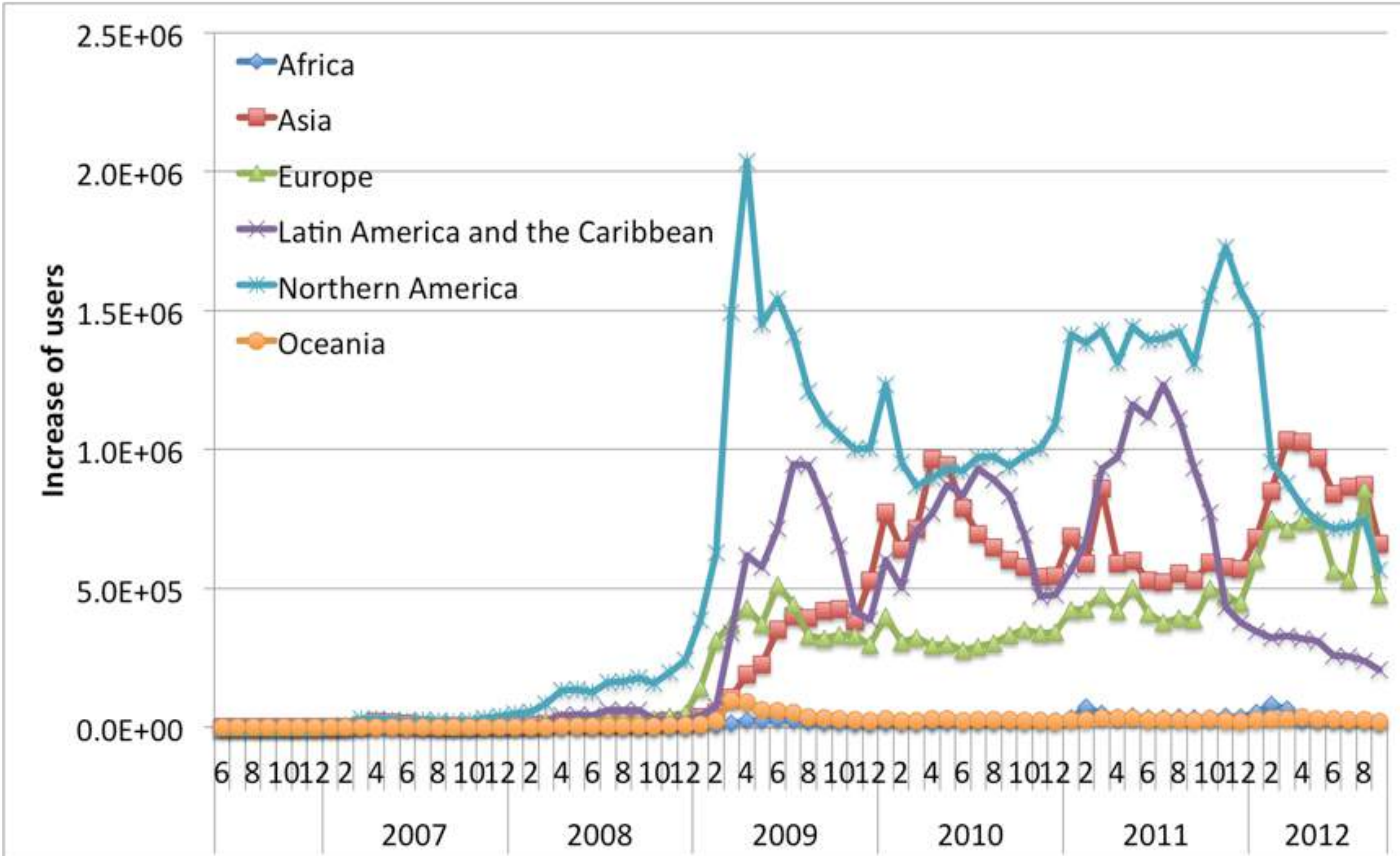
Characteristic of Twitter network also change?



Monthly increase of users by region

	July 2009		October 2012	
	# users	ratio (%)	# users	ratio (%)
Africa	0.13M	0.66	1.27M	0.96
Asia	1.65M	**8.30**	27.4M	20.8
Europe	3.01M	15.1	19.8M	15.1
Latin	3.80M	19.0	28.5M	21.6
NA	10.9M	**54.6**	53.1M	40.4
Oceania	0.45M	2.29	1.52M	1.15
Total	19.9M	100	131M	100

Monthly Increase of Users by Regions



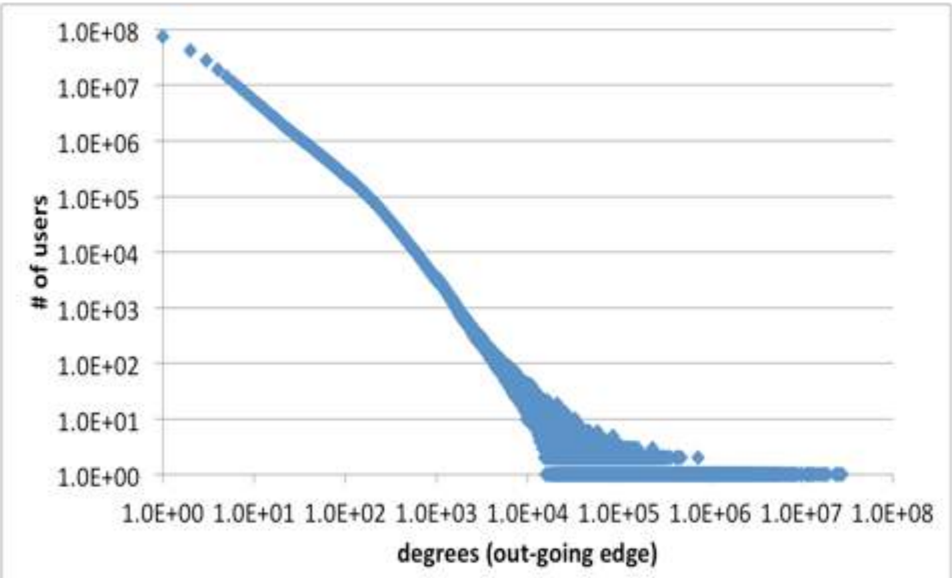
Degree Distribution: Unexpected value in in-degree distribution

"Scale-free" is one of the features of a social graph

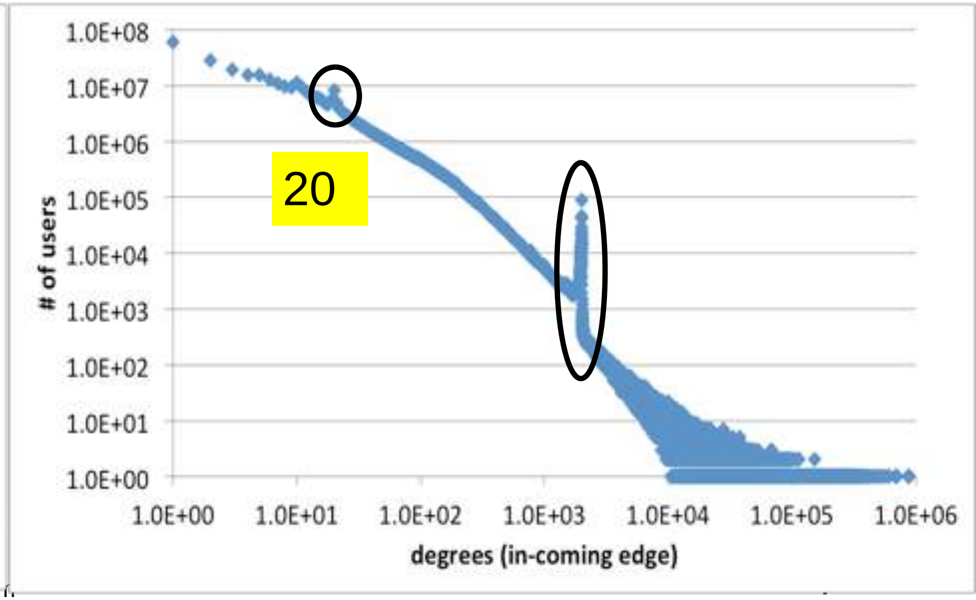
Unexpected value in in-degree distribution

- at x=20 due to Twitter recommendation system
- at x=2000 due to upper bound of friends before 2009

Out-degree distribution (follower)



In-degree distribution (friend)



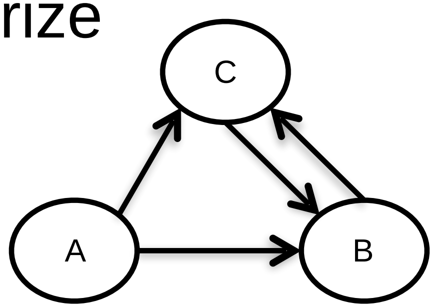
Reciprocity : decline from 22.1% to 19.5%

Reciprocity is a quantity to specifically characterize directed networks. Traditional Definition:

$$r = \frac{L^{\leftrightarrow}}{L}$$

$L^{\leftrightarrow}$: # of edges pointing in both directions

L : # of total edges



$L^{\leftrightarrow} = 1$

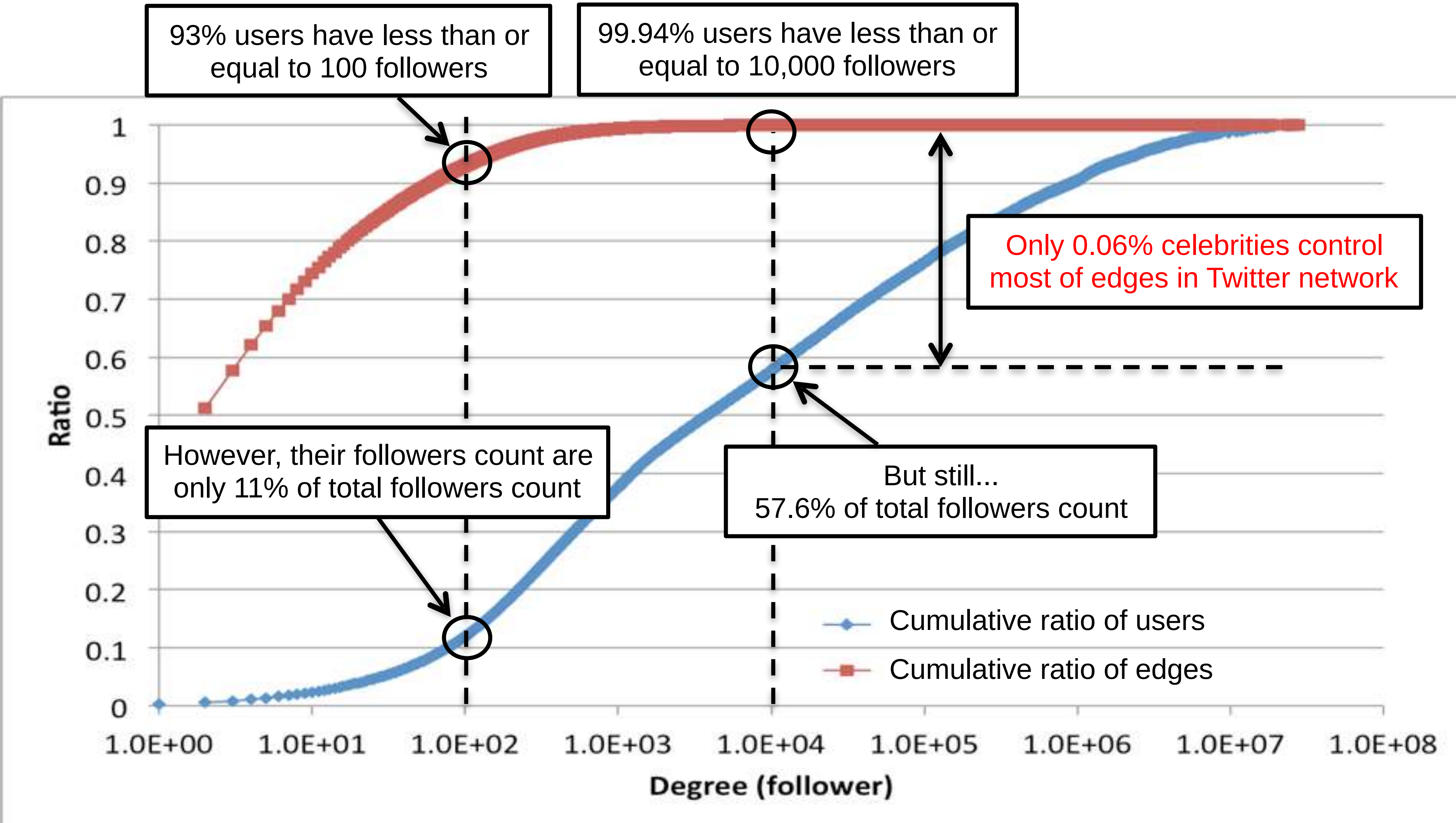
$L = 3$

- As a result, **Twitter network reciprocity decline from 22.1% to 19.5%**

	July 2009	October 2012
# of users	41.6 M	465.7 M
# of edges	1.47 B	28.7 B
Reciprocity	22.1% *1	19.5%

*1 : “What is Twitter, a social network or a news media?”

How many edges do celebrities have in Twitter network ? → Only 0.06% celebrities control most of edges



Degree of Separation and Network Diameter (1/3)

Both degree of separation and diameter are measures to characterize networks in terms of scale of graph.

Definition

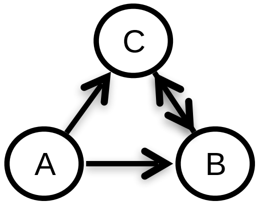
Degree of Separation:

Average value of the shortest-path length of all pairs of users.

Diameter:

Maximum value of the shortest-path length of all pairs of users

* Note : unreachable pairs are excluded from calculation



(A, B) = 1
(A, C) = 1
(B, A) = ∞
(B, C) = 1
(C, A) = ∞
(C, B) = 1

➡

Degree of Separation : 1
Diameter : 1

Degree of Separation and Network Diameter (2/3)

Experimental environment

- Using an approximate algorithm named **HyperANF [Paolo, WWW'12]** on TSUBAME 2.0 (Supercomputer at TITECH)
  - TSUBAME 2.0 Fat node
    - **64 cores, 512 GB memory**, SUSE Linux Enterprise Server 11 SP1
  - HyperANF Parameters
    - We set the logarithm of the number of registers per counter to 6 in order to reduce an error.

- Four times executions
  - Degree of Separation
    - take a average of 4 calculation
  - Diameter
    - take a minimum value of 4 calculation
    - because HyperANF guarantee lower bound of diameter
  - **Each execution on 2012 took more than 42,000 sec.**

Degree of Separation and Network Diameter (3/3)

Degree of Separation

- Only a little difference between '09 and '12 in spite of the lapse of three years.

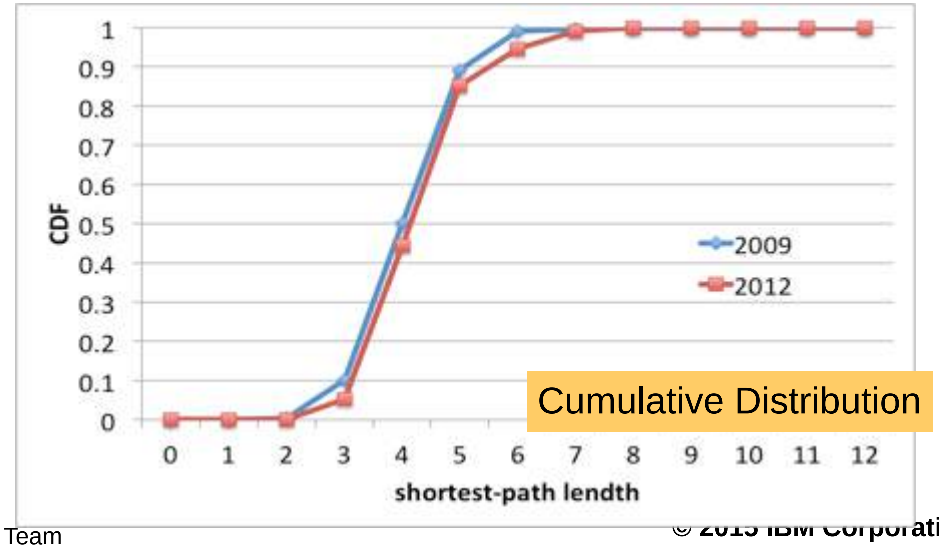
Diameter

- Diameter of 2012 is much larger than the one of 2009.

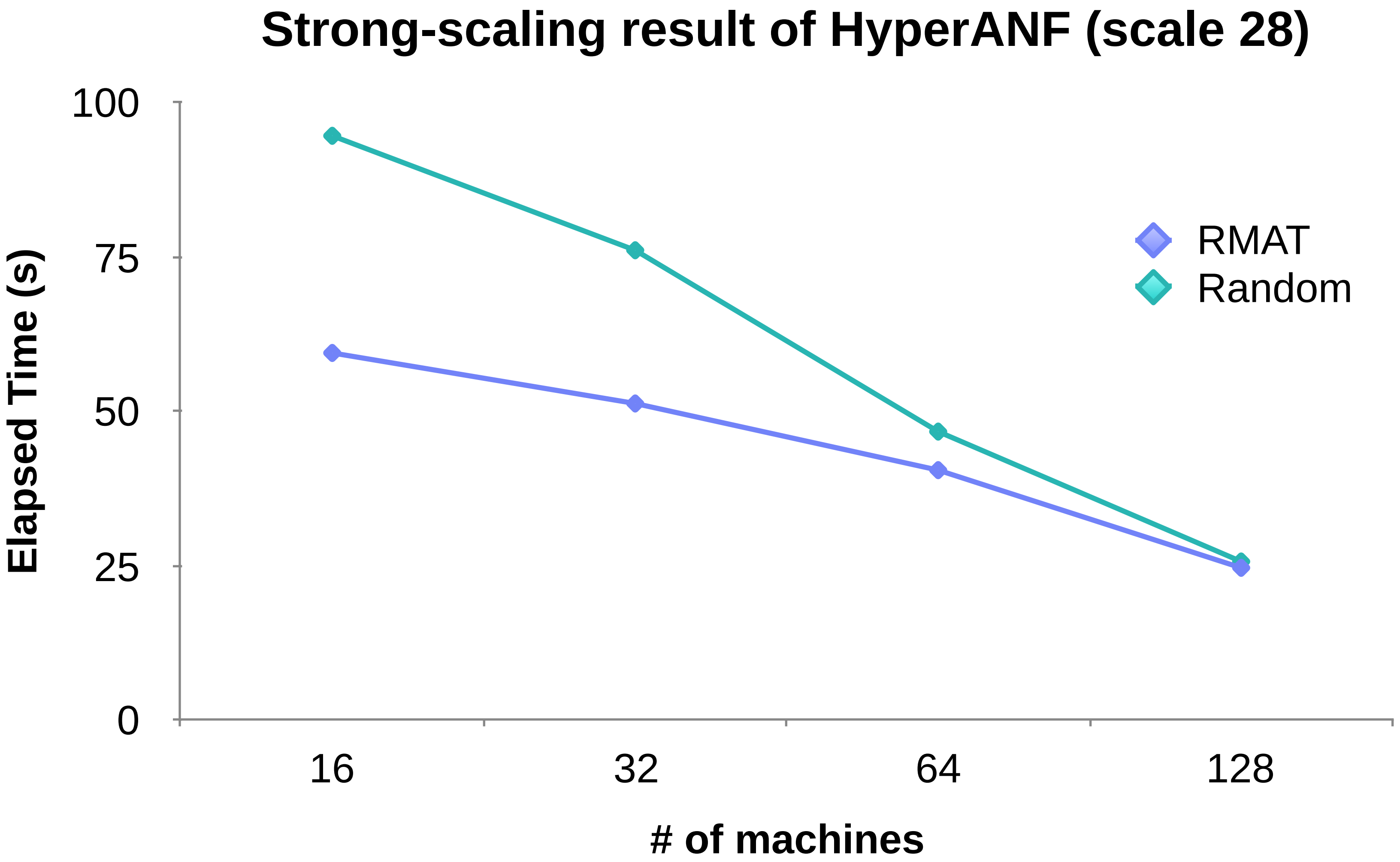
Cumulative Distribution

- In 2009
  - 89.2% of node pairs whose path length is 5 or shorter
  - 99.1% pairs whose it is 6 or shorter.
- In 2012
  - 85.2% pairs whose it is 5 or shorter
  - 94.6% pairs whose it is 6 or shorter

	Degree of Separation		Diameter	
	2009	2012	2009	2012
1st	4.39	4.48	25	70
2nd	4.46	4.65	26	71
3rd	4.53	4.54	25	70
4th	4.62	4.71	25	71
Result	**4.50**	**4.59**	**26**	**71**

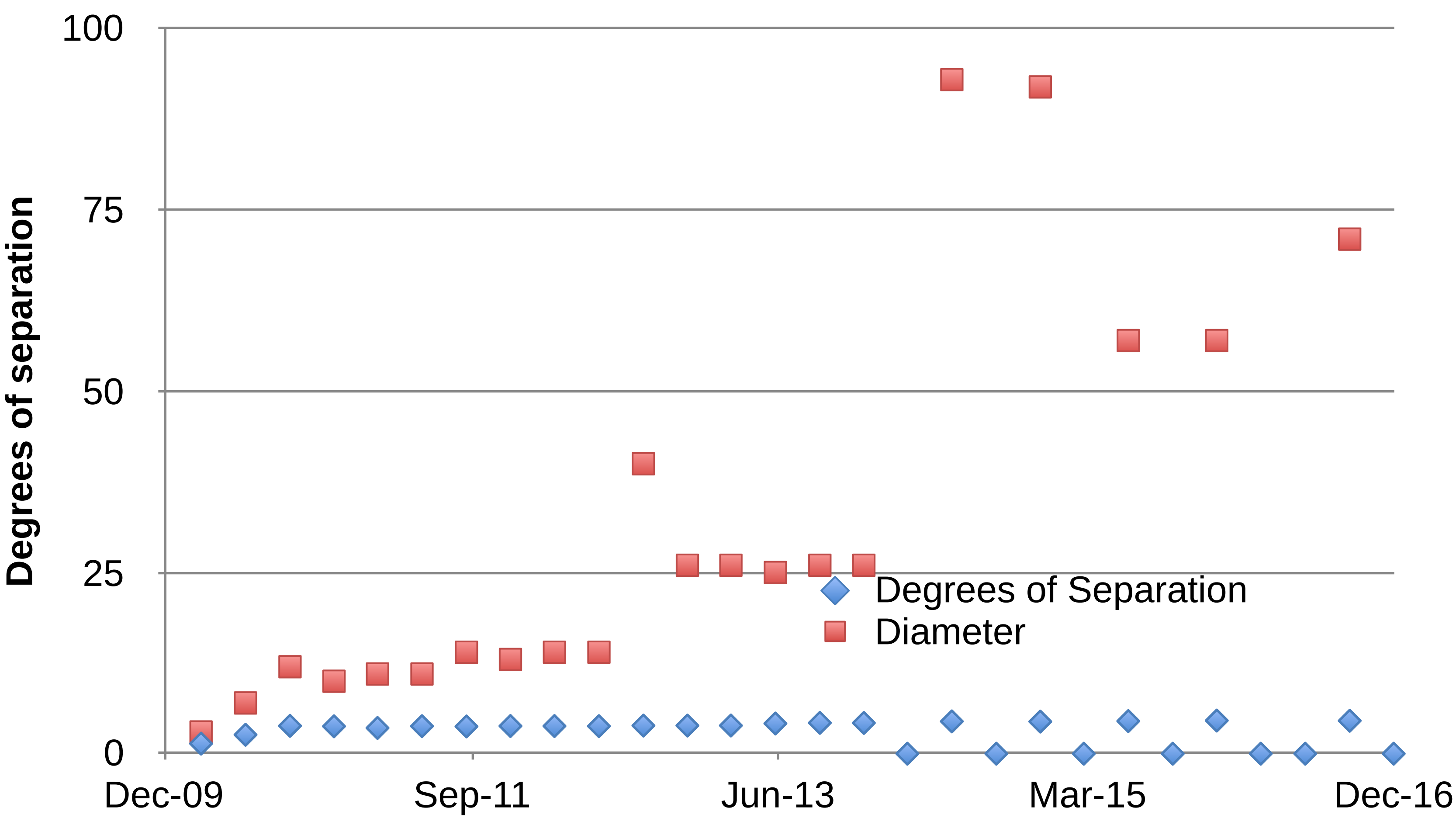


Computing Degree of Separation with ScaleGraph on Distributed Systems



The scale-28 graphs we used have 2^{28} (≈268 million) of vertices and 16×2^{28} (≈4.29 billion) of edges

Degree of Separation and Diameter for Time-Evolving Twitter Network



Classifying Degree of Separation by Spoken Language

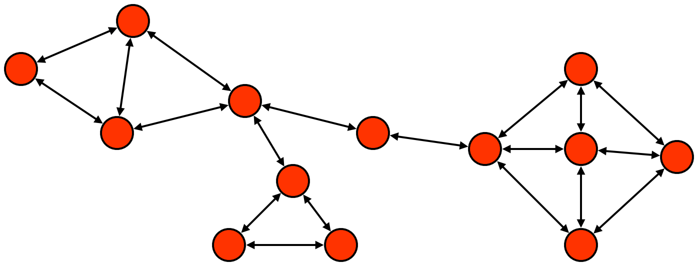
	Spanish	Portuguese	Japanese	Turkish	French
# of Users	64,927,267	22,456,938	20,279,402	10,402,846	10,743,511
Follow ratio to its own language	64%	58%	89%	57%	51%
Follow ratio to English	31%	36%	9%	39%	44%
# of Nodes for DOS	60,708,434	21,152,308	19,682,116	9,638,906	8,964,888
# of Edges for DOE	2,266,838,184	1,098,723,999	1,394,986,423	271,513,323	177,419,512
Average Degree	37.33	51.94	70.87	28.16	19.79
Degree of Separation (Average path length between two users)	4.625	4.253	4.014	4.340	4.699
Diameter（Lower bound value）	42	23	27	39	22

Q & A

Backup

- A graph:

$$G = (V, E)$$



- V = Vertices or Nodes
- E = Edges or Links

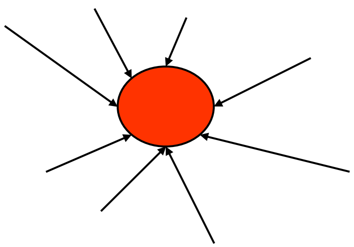
- The number of vertices: "Order" $N_v = |V|$

- The number of edges: "Size" $N_e = |E|$

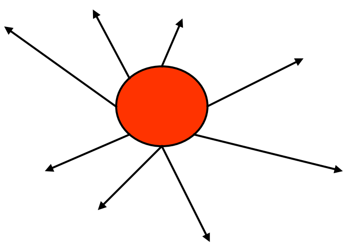
95

In-degrees and out-degrees

- For Directed graphs:



In-degree = 8

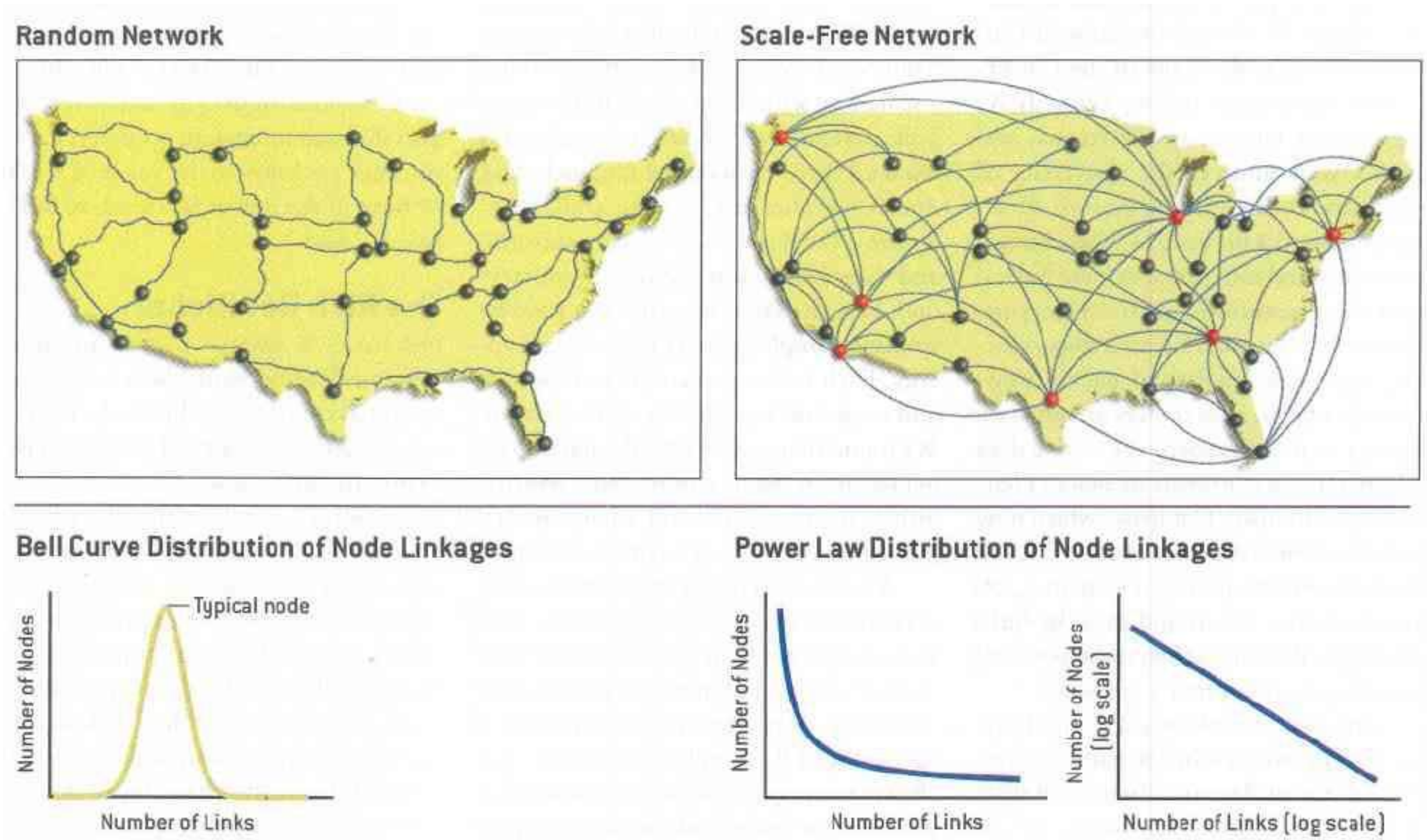


Out-degree = 8

96

Degree Distribution Example: Power-Law Network

A. Barbasi and E. Bonabeau, “Scale-free Networks”, Scientific American 288: p.50-59, 2003.



$$p_k = e^{-m} \cdot \frac{m^k}{k!}$$

$$p_k = C \cdot k^{-\tau} e^{-k/\kappa}$$

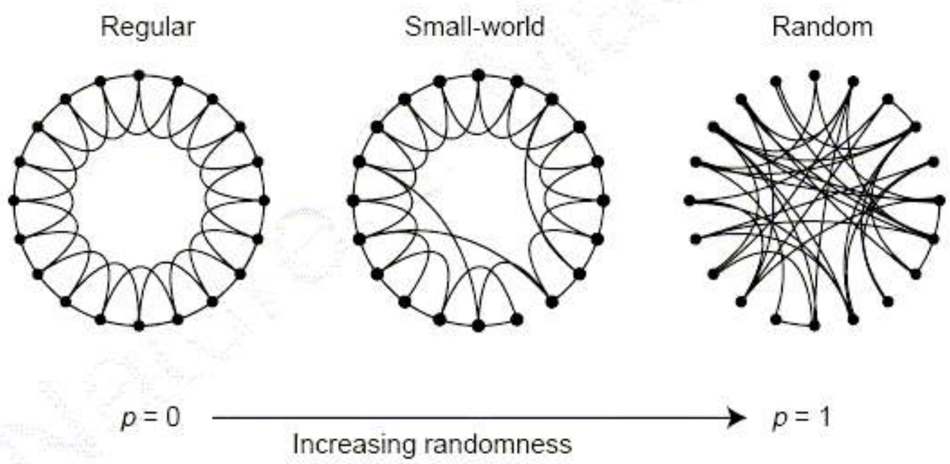
Newman, Strogatz and Watts, 2001

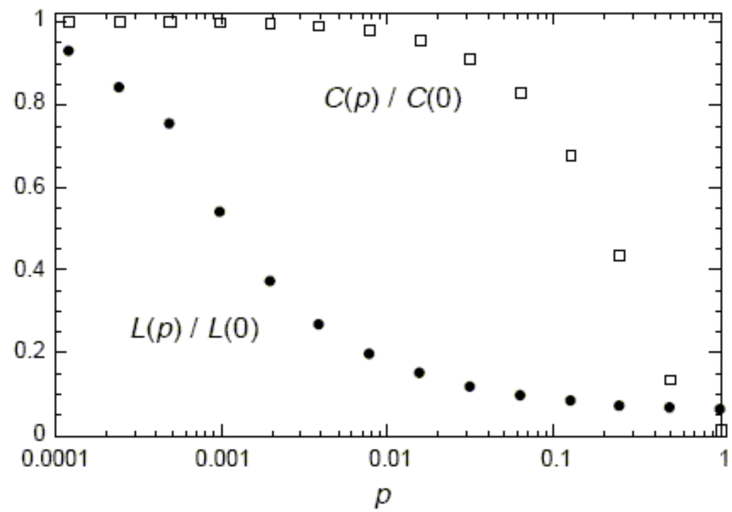
97

Six Degree Separation:

adding long range link, a regular graph can be transformed into a small-world network, in which the average number of degrees between two nodes become small.

from Watts and Strogatz, 1998



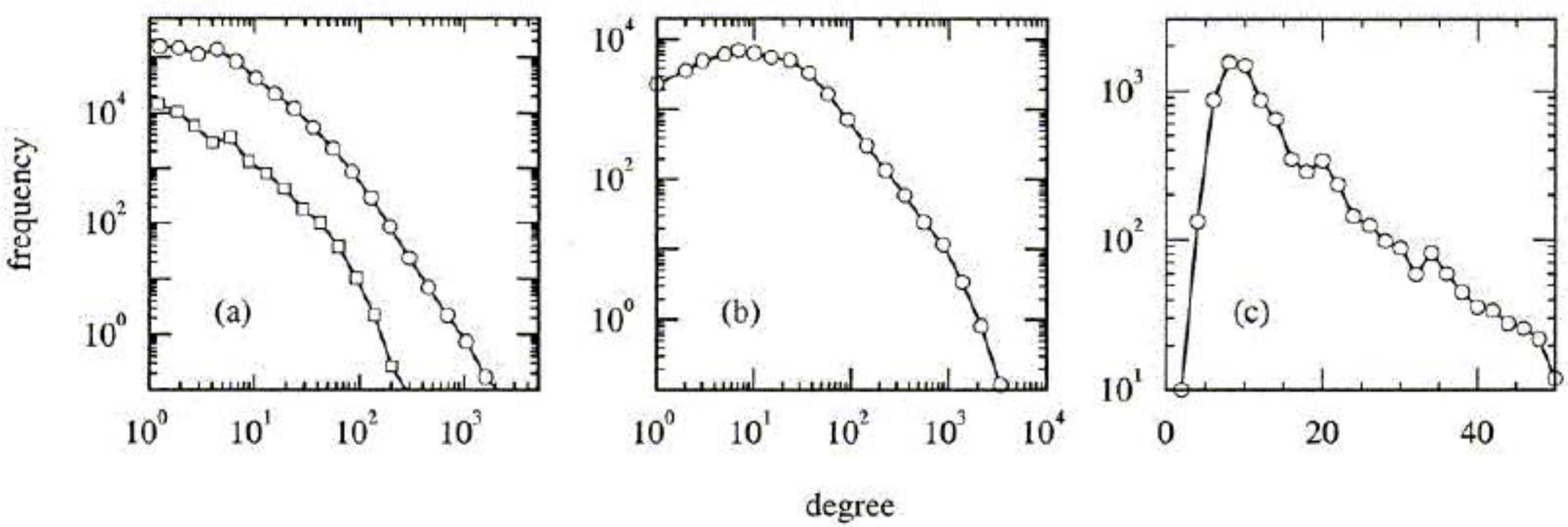


C: Clustering Coefficient, L: path length,
(C(0), L(0)): (C, L) as in a regular graph
(C(p), L(p)): (C,L) in a Small-world graph with randomness p.

98

Some examples of Degree Distribution

(a) scientist collaboration: biologists (circle) physicists (square), (b) collaboration of move actors, (d) network of directors of Fortune 1000 companies

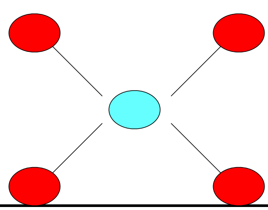


99

Relationship Between Network Topology and Productivity

- Network size is positively correlated with performance.
  - Each person in your email address book at work is associated with $948 dollars in annual revenue.

1 direct contact in a person's network



$74.07 *increase* in monthly revenues or $948 annual revenues

Std error =(26.38)***

Significant at $p < 0.01$

100

Centrality

"There is certainly no unanimity on exactly what centrality is or its conceptual foundations, and there is little agreement on the procedure of its measurement." – Freeman 1979.

Degree (centrality)

Closeness (centrality)

Betweeness (centrality)

Eigenvector (centrality)

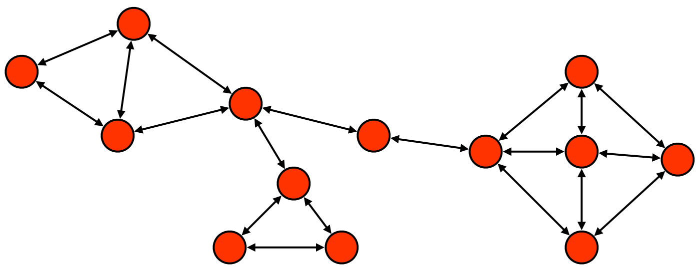
101

Closeness

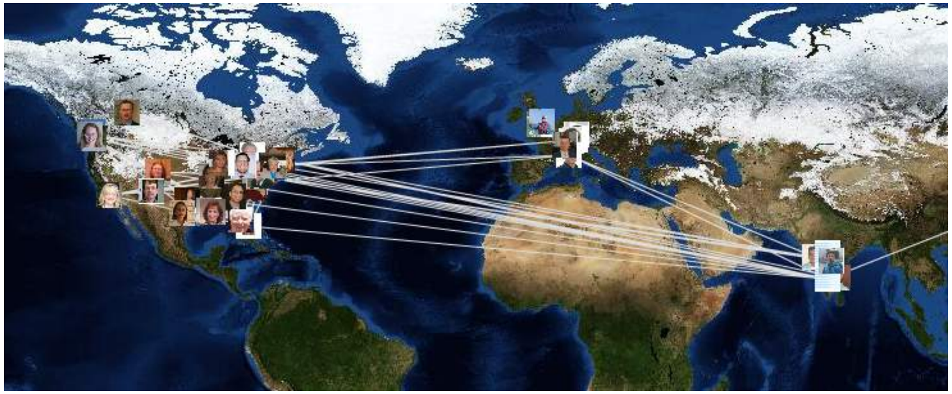
Closeness: A vertex is 'close' to the other vertices

$$c_{CI}(v) = \frac{1}{\sum_{u \in V} dist(v,u)}$$

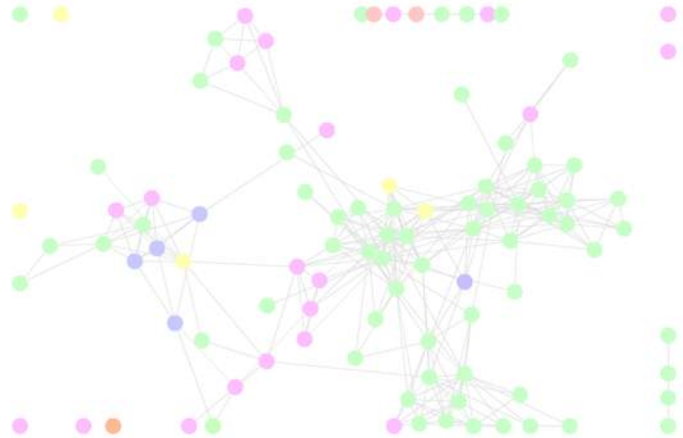
where dist(v,u) is the geodesic distance between vertices v and u.



Betweenness ==> Bridges

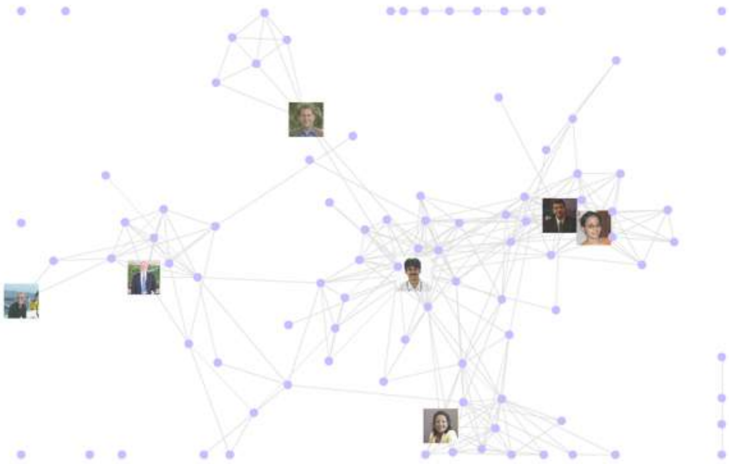


Example: Healthcare experts in the world



Connections between different divisions

Example: Healthcare experts in the U.S.



Key social bridges

103

Betweenness

Betweenness measures are aimed at summarizing the extent to which a vertex is located 'between' other pairs of vertices.

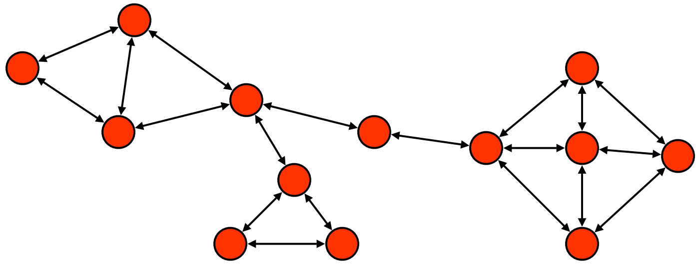
Freeman's definition:

$$c_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma(s,t \mid v)}{\sigma(s,t)}$$

Calculation of all betweenness centralities requires

calculating the lengths of shortest paths among all pairs of vertices

Computing the summation in the above definition for each vertex



Eigenvector Centrality

Try to capture the ‘status’, ‘prestige’, or ‘rank’.

More central the neighbors of a vertex are, the more central the vertex itself is.

$$c_{Ei}(v) = \alpha \sum_{\{u,v\} \in E} c_{Ei}(u)$$

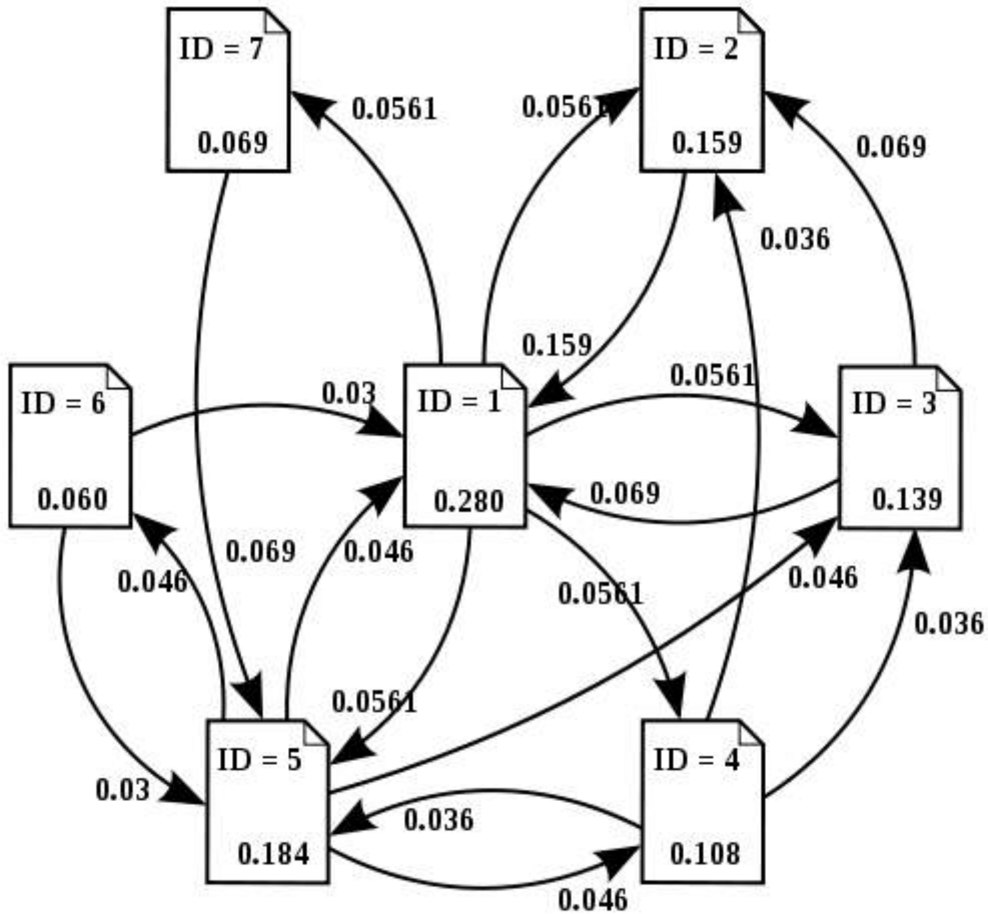
The vector $\mathbf{c}_{Ei} = (c_{Ei}(1), \ldots, c_{Ei}(N_v))^T$ is the solution of the

eigenvalue problem:

$$\mathbf{A} \cdot \mathbf{c}_{Ei} = \alpha^{-1} \mathbf{c}_{Ei}$$

105

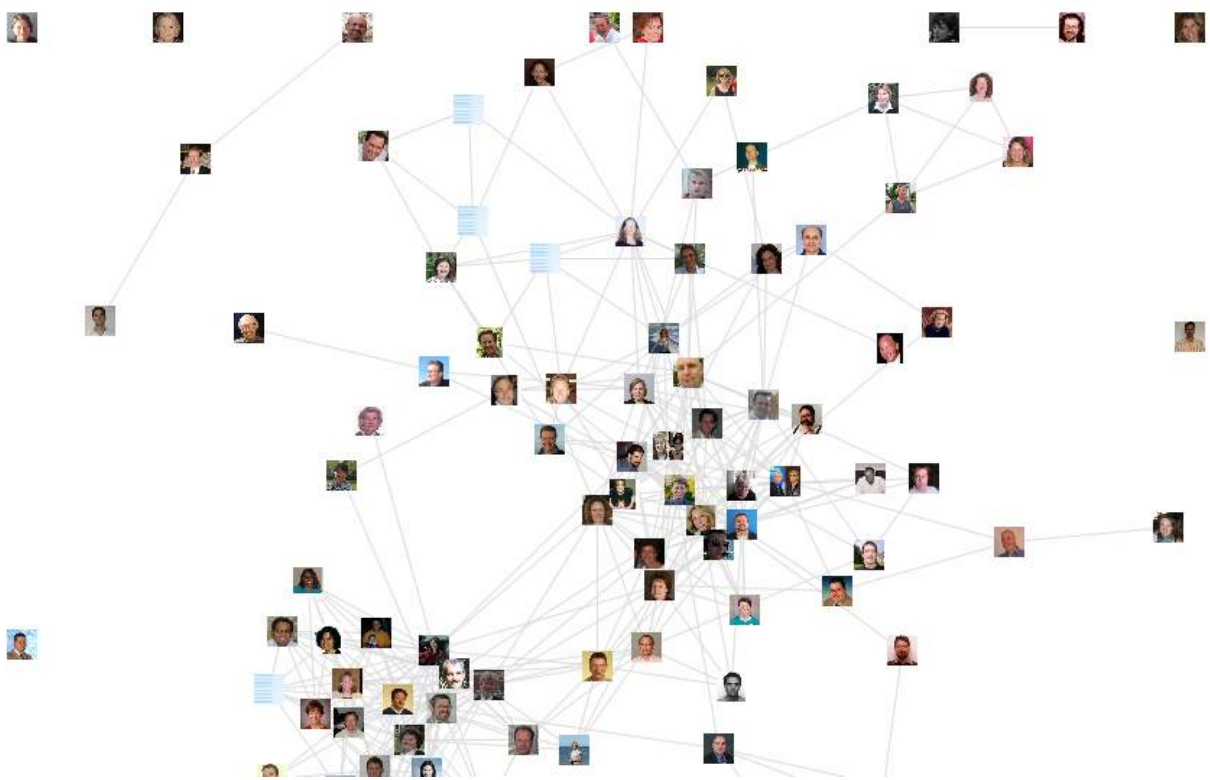
PageRank Algorithm (Simplified)



106

Connectivity of Graph

- A measure related to the flow of information in the graph
- Connected ➔ every vertex is reachable from every other
- A connected component of a graph is a maximally connected subgraph.
- A graph usually has one dominating the others in magnitude ➔ giant component.



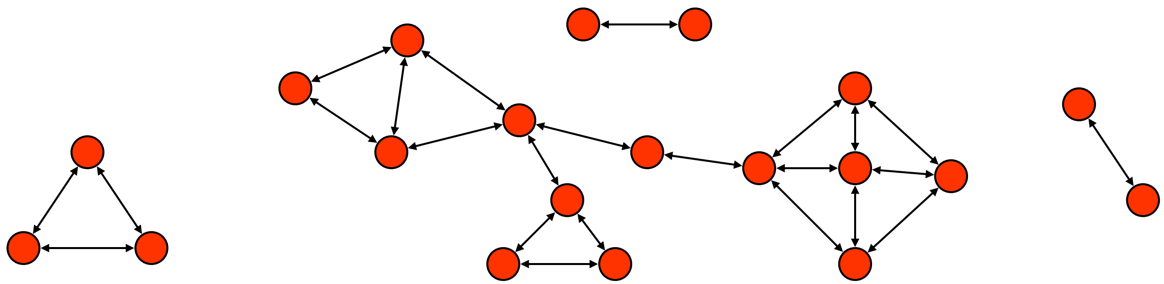
107

Reachable, Connected, Component

- Reachable: A vertex v in a graph G is said to be reachable from another vertex u if there exists a walk from u to v.

- Connected: A graph is said to be connected if every vertex is reachable from every other.

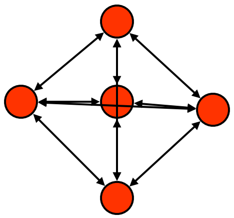
- Component: A component of a graph is a maximally connected subgraph.



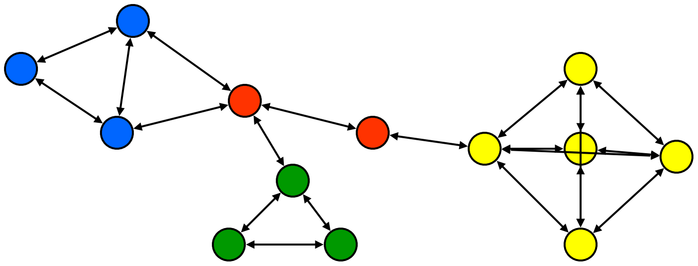
108

Families of Graphs

- Complete Graph: every vertex is linked to every other vertex.



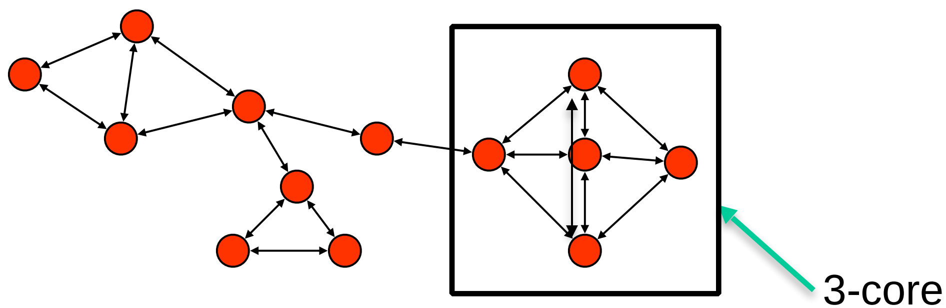
- Clique: a complete subgraph.



109

Another Weakened Versions of Cliques -- Cores

A k-core of a graph G is a subgraph H in which all vertices have degree at least k.



Batagelj et. al., 1999. A maximal k-core subgraph may be computed in as little as O(Nv + Ne) time.

Computes the shell indices for every vertex in the graph

Shell index of v = the largest value, say c, such that v belongs to the c-core of G but not its (c+1)-core.

For a given vertex, those neighbors with lesser degree lead to a decrease in the potential shell index of that vertex.

110

The density of a subgraph H = (VH , EH) is:

$$den(H) = \frac{|E_H|}{|V_H|(|V_H|-1)/2}$$

Range of density

$$0 \leq den(H) \leq 1$$

and

$$den(H) = (|V_H|-1)\bar{d}(H)$$

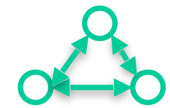
average degree of H

Clustering coefficient

A triangle is a complete subgraph of order three.

A connected triple is a subgraph of three vertices connected by two edges (regardless how the other two nodes connect).

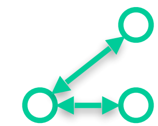
The local clustering coefficient can be expressed as:



$$den(H_v) = cl(v) = \frac{\tau_\Delta(v)}{\tau_3(v)}$$

# of triangles

# of connected triples for which 2 edges are both incident to v.



The clustering coefficient of G is then:

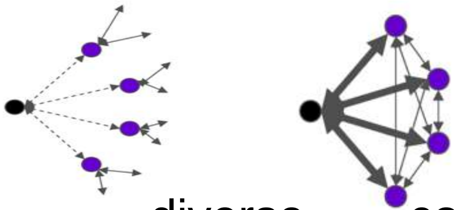
$$cl(G) = \frac{1}{V'} \sum_{v \in V'} cl(v)$$

Where $V' \subseteq V$ is the set of vertices v with $dv \geq 2$.

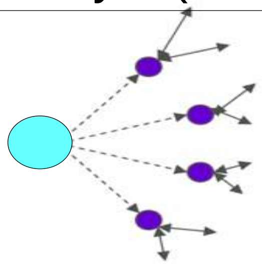
112

Which one is better -- Structural Diverse Networks or Cohesive Networks?

- Structural diverse networks with abundance of structural holes are associated with higher performance.
  - When friends of your friends are not friends of each other or belong to the same social group.



1 standard deviation increase
in network diversity
Diversity = (1-constraint)



→

276.64 % ***increase*** **in monthly revenues**
Std error =(113.88)
Significant at $p < 0.01$

113

ROI – Productivity Increase after User Adopting SmallBlue

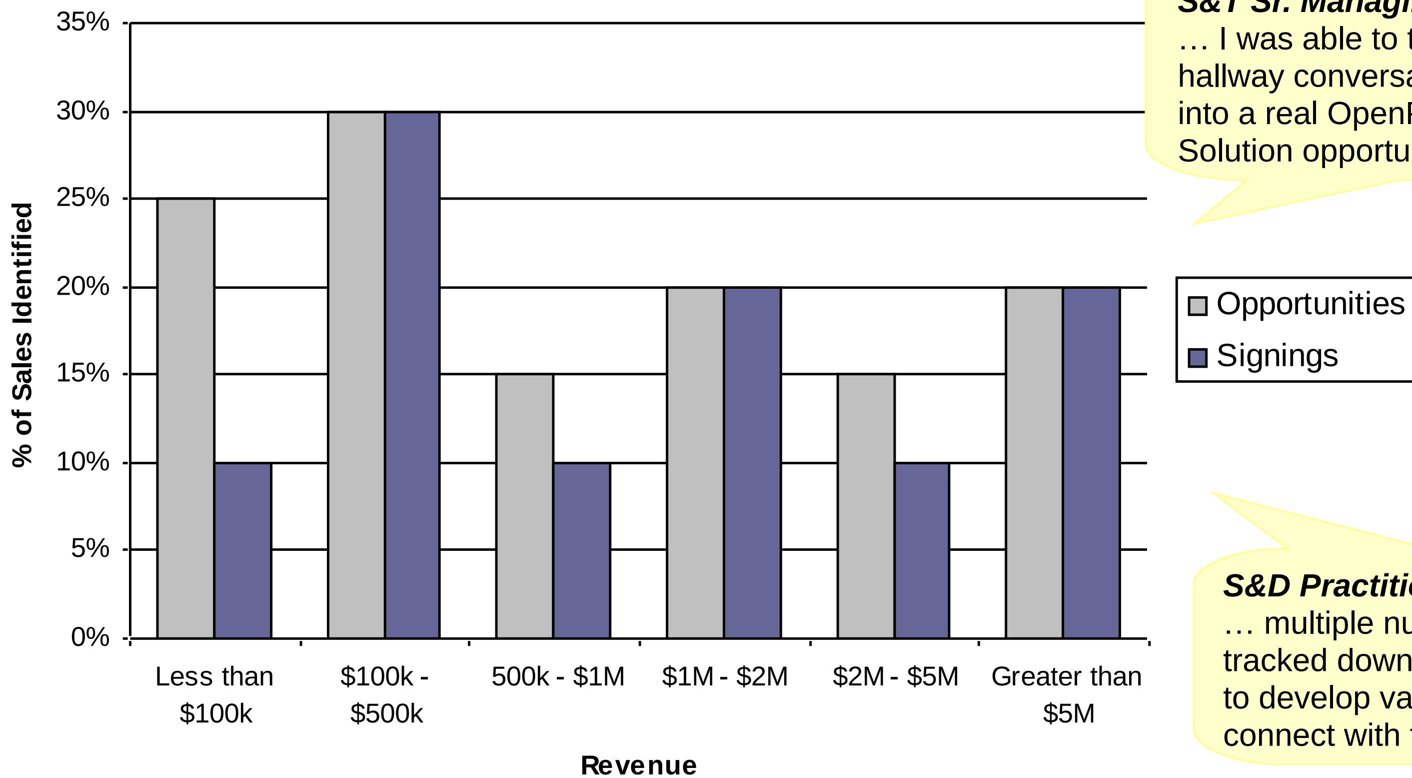
Studied 2,038 global practitioners for 2 years by Prof. Wu, Wharton School, U Penn.

Controlled factors: temporal shocks, individual characteristics such as job roles and hierarchies, and the characteristics of each project including line of business and the region.

Compare individual performance == billable revenue since adopting SmallBlue.

We saw a revenue of $584.15 per month == $7,010 in a year

ROI – Sales opportunities & signings as a result of using SmallBlue SNA tool



S&T Sr. Managing Consultant, US
… I was able to turn a 2 minute hallway conversation with my client into a real OpenPages Compliance Solution opportunity!

S&D Practitioner, India
… multiple number of times I tracked down assets across SWG to develop value propositions and connect with the right expert

Based on 324 random surveys in 2011 (0.35% sampling rate):

- $40M in unqualified sales opportunities identified
- $9M in identified sales based on unqualified opportunities
- 5% of the sample pool of survey respondents realized revenue
- All achieved through using SmallBlue

Studied by IBM
Global Business Services