CS 8803
Big Data Systems and Analytics

Ling Liu

Professor
School of Computer Science
College of Computing, Georgia Tech

Lecture 1 Outline

- A Brief Review of Syllabus
  - Lecture, Grading Scheme, Projects, Exams
- Big Data Systems and Analytics: a Strawman Argument
  - Is it simple?
  - Why is it not?
  - What are the important issues?
  - Topics to be covered in this course

Course Information

- Instructor
  - Ling Liu
  - lingliu@cc.gatech.edu
  - Office: KACB 3340
  - Office hours
- TA:
  - Abdurrahman Yasar (ayasar@gatech.edu)
  - Emre Yigitoglu (eyigitoglu@gatech.edu)

Course Objectives

- Review and introduce
  - concepts, techniques, algorithms and systems issues in big data systems and analytics
  - Cover the fundamentals for data science education
- Understand and explore
  - Cutting Edge Big Data Technology and R & D areas
  - New technical challenges, open research issues, which are critical for developing Big Data Systems, Algorithms and Applications

Course Structure

- Lectures
  - by Instructor + invited guest speakers
  - Discussions and interaction in the class
- Your Part
  - Homework (4)
    - Reading + writing critiques
    - Programming
  - Project (team):
    - proposal, implementation, demo, workshop presentation
  - Technology Review (final exam)
  - Participation in the class

Administravia

- Office Hours
  - Thursdays 11am -12noon, or by appointment
- Grading
  - Class Participation 15%
  - Homework Assignments (5) 20%
  - Project 50%
  - Final (Technology Review) 15 %
- Announcements
  - In class and on T-Square
- Course Materials
  - No textbooks Required
  - Course Notes + A collection of papers, available on Tsquare
- Useful and Related Links
  - TSquare postings under resource

Homework Assignments

- 4 homework assignments (starting from the 3rd week)
  - Posted on Monday of Week 3, 6, 8, 10
  - Due on Friday of Week 4, 7, 9, 11
- All homework assignments are individual and should be completed independently
- Two Types of Homework Assignments (your choice)
  - Reading Assignment
  - Programming Assignment

Reading Assignment

- For each reading assignment, you choose two papers from the list of recommended readings.
- Two papers should be related to one subject.
- You are asked to write two reading critiques, one for each of the two papers you read.
- Submit your reading summary on TSquare by the due date.

Suggestions and Tips for Reading Summary

- Read each paper at least twice
- Comments on the following questions:
  - What is the subject area?
  - What are the general problems and specific problems this paper intends to address
  - Describe the main ideas and techniques of the paper
  - Summarize the strong and weak points
  - Does the paper solve the problems promised?
  - Is there any other solutions?
  - What are your suggestions for revision and/or what are your insights for the problem addressed?

Reading Summary Template

CS8803/CS4365 Homework Reading Summaries

Student ID:

Family Name:

Given Name:

Paper Title:

Summary/Critique

(1) Problems

(2) New Idea and Strengths

(3) Weaknesses and Extensions

Reading Summary - Grading Scheme

- Grading scale: 0-100 points
- Pass (>=60)
  - a good summary to the assigned reading
- Pass – (<60)
  - in limited circumstances in which either the summary is incomplete or it is clear from the summary that the student did not read the material or the summary contains poorly worded criticism.
- Pass + (>90)
  - a summary in which the student goes beyond a basic review and presents insight that is thoughtful and well expressed.

Programming Assignment

- For each programming assignment, you are required to choose one task from the list of recommended programming tasks.
- The Programming Assignment should be submitted to Tsquare by the due date
- Deliverable includes the following 4 items
  - Source code + Read Me (or open source URL)
  - Executable code
  - Inputs and outputs of your program (e.g., datasets used)
  - Flow Chart of your program

    https://en.wikipedia.org/wiki/Flowchart

    http://users.evtek.fi/~jaanah/IntroC/DBeech/3gl_flow.htm

Programming Assignment: Grading Scheme

- Grading scale: 0-100
- Pass (>=60)
  - a complete package of deliverable with quality documentation
- Pass + (>= 90)
  - a complete package of deliverable with quality documentation
  - The program presents some novelty in design or implementation or the documentation is superb and GUI is elegant.
- Pass – (<60)
  - The program did not provide the full functionality as required
  - The documentation is incomplete or poorly written

Project

- Interesting, Novel, and value-added ideas/Concepts
- Innovative engineering/executions
- Four Components
  - Proposal (20%)
  - Workshop Project Presentation (20%)
  - Final Project Demo + Deliverables (60%)

Two Types of Projects

- Big Data Systems Projects
  - Novel in System Design, Optimization and Engineering
  - Novel in Problem Identification and/or Solution Approach
- Data Analytics Projects
  - Novel in Analytic Problems to be solved
  - Novel in Algorithm design and implementation methods (performance and analytic quality/accuracy)
  - Novel in Applications being deployed

Project

- Project team: 2-4 persons per team
- Count 50% of your total grade
- Four Types of Deliverables
  - Project Proposal
    - proposal submission, meeting with instructor, proposal revision
  - Project presentation at the workshop
    - Every team member must be present at the project presentation
  - Project Demo
    - Every team member must be present at the demo
  - Final project deliverable

Project (cont.)

- Project Requirement
  - Must be topics related to Big Data Systems or Application-specific Analytics
  - Make sure that your project exhibits innovative ideas or novel applications.
- Proposal Due on Friday of Week 5 (Sept 23)
  - 2-3 pages (in pdf or word)
  - covering problems to be addressed, concepts, techniques, system architecture, and the hardware/software environment to be used.
  - A statement with adequate elaboration on why the proposed project is innovative and useful

Project (cont.)

- Project team meeting with Professor
  - Tuesday of Week 6 (Sept 27)
  - Sign-up on TSquare course Wiki
  - Location: in the small conf. room across my office KACB 3340
- Selected project proposal presentation
  - Invitation in Week 6
  - Presentation on Tuesday of Week 7
- Proposal revision if required due by Friday of Week 8

Project (cont.)

- Final Project Presentation - Workshop
  - 15 ~ 20 minutes, incl. 5 min questions
  - emphasize on new ideas and the most interesting components of your project
  - Present evaluation methods and lessons learned
  - Every team member must present your contribution as a integral part of your team project.

Project (cont.)

- Final Project Demo
  - 15~20 minutes in the small conf. room across my office KACB 3340
  - What to hand in?
    - In-class project presentation (ppt), submit to TSquare
    - Project Deliverable, submit to TSquare
      - Final Report (ppt or word, highlights, figures, Screen shots)
      - Source Code with ReadMe
      - Executable (using .tar file, compressed)
      - Datasets used for measurement and demo (Input)
      - Data output + Performance measurements

Technology Review

- Topic Selection
  - Topics covered or highly related to the topics covered in the courses, incl. lectures and homework reading and programming assignments.
  - can be combined with the theme of your course project (optional).
- Structure
  - Should cover the state of art in the selected technology area.
  - should contain a discussion section describing your thought and your prediction on the deployment and adoption of the technology being reviewed.
  - Should provide a list of major references that you have used in preparing your review.
  - The expected length of the technology review is about 10~15 pages, single column and single spacing (1.2 pt). Figures are welcome.
  - Due date: Final exam day: **Dec 15 (Thu) 3pm**

Questionnaire

- Fill in the course questionnaire
  - Accessible on Tsquare for those who have registered the course
  - Upload your answer to Tsquare https://t-square.gatech.edu/

Lecture 1

- A Strawman Big Data System
  - Walk through the process of big data processing
- A Strawman Data Analytic Algorithm
  - Walk through the process of data analysis algorithm
- Discussion
  - Many problems with simple implementation
  - Topics Covered in this course

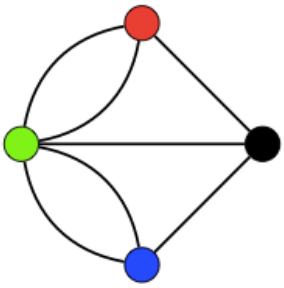
Graphs

- Real information networks are represented as graphs and analyzed using graph theory
  - Internet
  - Road networks
  - Utility Grids
  - Protein Interactions
- A graph is
  - a collection of binary relationships
  - viewed as networks of pairwise interactions between physical or conceptual entities

Scale of the first graph

Nearly 300 years ago the first graph problem consisted of 4 vertices and 7 edges—*Seven Bridges of Königsberg* problem.

[Paul Burkhardt, Chris Waring 2013]

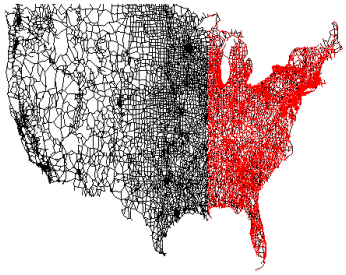


Crossing the River Pregel

Is it possible to cross each of the Seven Bridges of Königsberg exactly once?

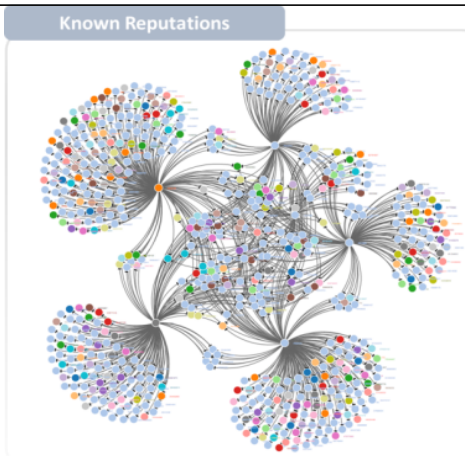
The data set and the algorithm can easily fit in the DRAM "memory"

Road Scale



>24 million vertices
>58 million edges
*Route Planning in Road Networks - 2008

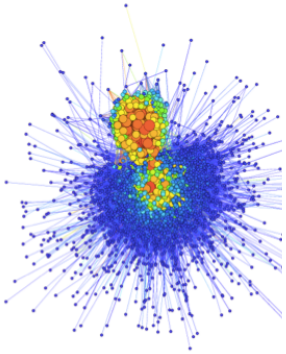
Social Scale

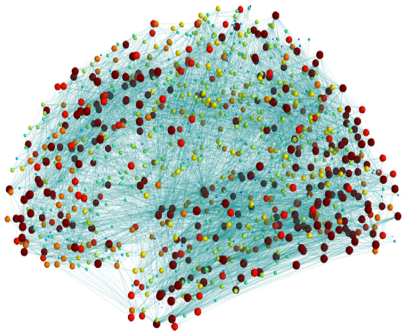


>1 billion vertices
~ 1 trillion edges
*Facebook Engineering Blog

~41 million vertices
>1.4 billion edges
*Twitter Graph- 2010

Web Scale

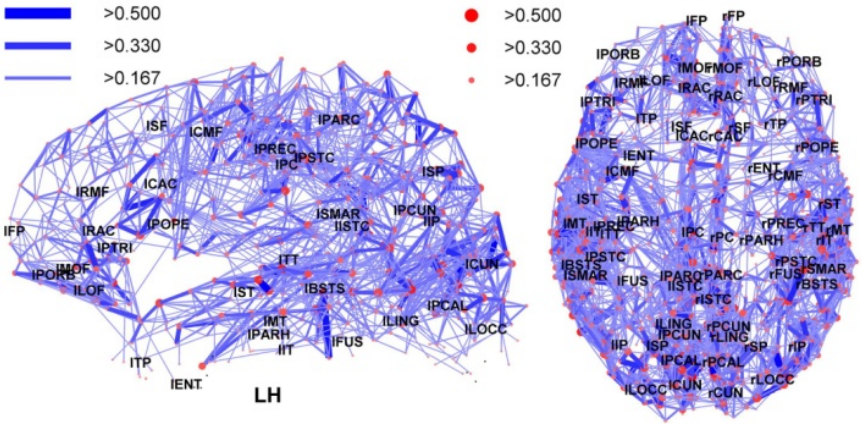




bing

>50 billion vertices
>> 1 trillion edges
*NSA Big Graph Experiment- 2013

Brain Scale



>100 billion vertices
>100 trillion edges
*NSA Big Graph Experiment- 2013

Scale of Today's Graphs

- **Scale of Graphs studied in current CS literature**: on the order of billions of edges, tens of GBs

[Paul Burkhardt, Chris Waring 2013]

Popular graph datasets in current literature

	n (vertices in millions)	m (edges in millions)	size
AS-Skitter	1.7	11	142 MB
LJ	4.8	69	337.2 MB
USRD	24	58	586.7 MB
BTC	165	773	5.3 GB
WebUK	106	1877	8.6 GB
Twitter	42	1470	24 GB
YahooWeb	1413	6636	120 GB

AS by Skitter (AS-Skitter) — internet topology in 2005 (n = router, m = traceroute)
LiveJournal (LJ) — social network (n = members, m = friendship)
U.S. Road Network (USRD) — road network (n = intersections, m = roads)
Billion Triple Challenge (BTC) — RDF dataset 2009 (n = object, m = relationship)
WWW of UK (WebUK) — Yahoo Web spam dataset (n = pages, m = hyperlinks)
Twitter graph (Twitter) — Twitter network[1] (n = users, m = tweets)
Yahoo! Web Graph (YahooWeb) — WWW pages in 2002 (n = pages, m = hyperlinks)

Real World Graphs: How Big?

1 terabyte (TB) =1,024GB ~10^3GB
1 petabyte (PB) =1,024TB ~ 10^6GB
1 exabyte (EB) =1,024PB~10^9GB
1 zettabyte (ZB) =1,024EB~10^{12}GB.
1 yottabyte (YB) =1,024ZB~10^{15}GB.

Social Scale

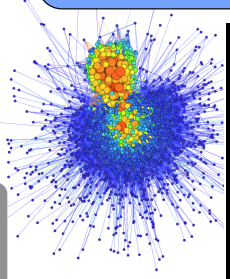
- 1 billion vertices, 100 billion edges
  - 111 PB adjacency matrix
  - 2.92 TB adjacency list

Web Scale

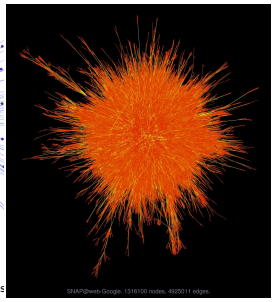
- 50 billion vertices, 1 trillion edges
  - 271 EB adjacency matrix
  - 29.5 TB adjacency list

Brain Scale

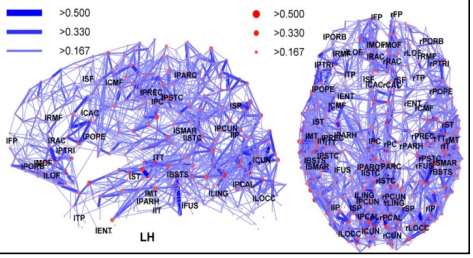
- 100 billion vertices, 100 trillion edges
  - 1.1 ZB adjacency matrix
  - 2.83 PB adjacency list



Twitter graph from Gephi datas
(http://www.gephi.org)



Web graph from the SNAP database
(http://snap.stanford.edu/data)



How do we store and process graphs?

- Conventional approach is to store and compute graph analysis in-memory.
  - **SHARED-MEMORY**
    - Parallel Random Access Machine (PRAM)
    - data in globally-shared memory
    - implicit communication by updating memory
    - fast-random access
  - **DISTRIBUTED-MEMORY**
    - Bulk Synchronous Parallel (BSP)
    - data distributed to local, private memory
    - explicit communication by sending messages
    - easier to scale by adding more machines

[Paul Burkhardt, Chris Waring 2013]

Top Super Computer Installations

Largest supercomputer installations do not have enough memory to process the Brain Graph (3 PB)!

- Titan Cray XK7 at ORNL — #1 Top500 in 2012
  - 0.5 million cores
  - 710 TB memory
  - 8.2 Megawatts[3]
  - 4300 sq.ft. (NBA basketball court is 4700 sq.ft.)
- Sequoia IBM Blue Gene/Q at LLNL — #1 Graph500 in 2012
  - 1.5 million cores
  - 1 PB memory
  - 7.9 Megawatts[3]
  - 3000 sq.ft.

1 terabyte (TB) =1,024GB ~10^3GB
1 petabyte (PB) =1,024TB ~ 10^6GB
1 exabyte (EB) =1,024PB~10^9GB
1 zettabyte (ZB) =1,024EB~10^{12}GB

Brain Scale

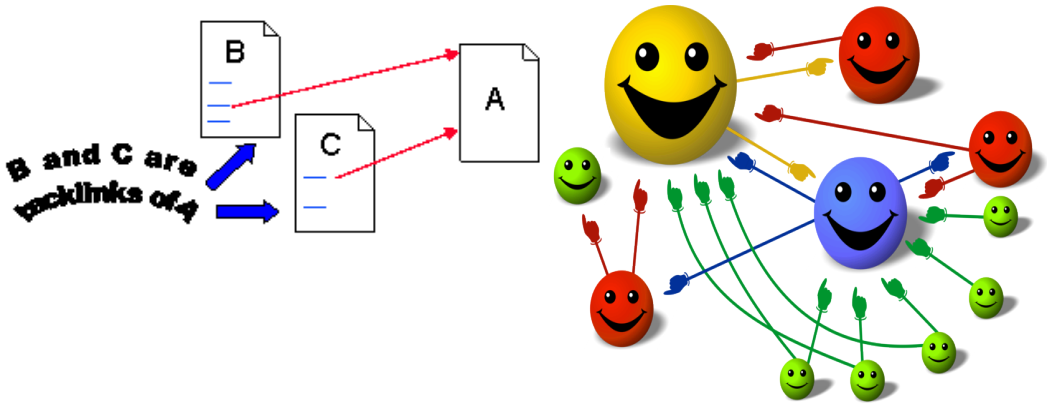
- 100 billion vertices, 100 trillion edges
  - 1.1 ZB adjacency matrix
  - 2.83 PB adjacency list

Graph Computation: How hard can it be?

- **Algorithm complexity really matters!**
  - Run-time of $O(n^2) \sim O(n^m)$ on a trillion node graph
- **Increased size imposes greater challenge . . .**
  - Latency
  - resource contention
- **Difficult to parallelize (data/computation)**
  - irregular data access increases latency
  - skewed data distribution creates bottlenecks
    - high degree vertices
    - highly skewed edge weight distribution
    - giant component v.s. tiny component

Google PageRank

- Quality of a page is related to its in-degree
- Ranking pages based on links to them
  - Links from many pages → high rank
  - Link from a high-rank page → high rank

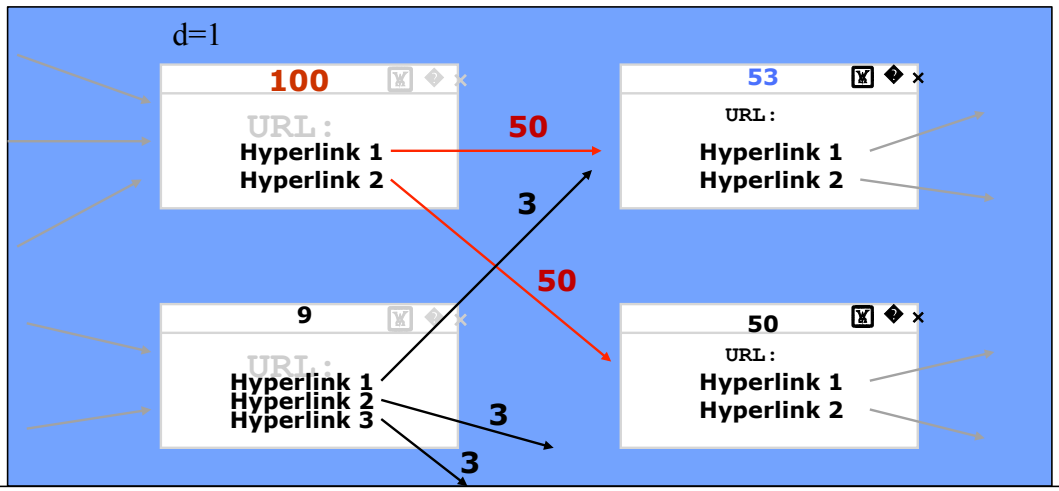


PageRank: An Example

- Simplified Definition of PageRank

$$R(u) = d \sum_{v \in B_u} \frac{R(v)}{N_v}$$

F_u : the set of pages u points to
B_u : the set of pages that point to u
N_u = | F_u |
d: damping factor (0.1]



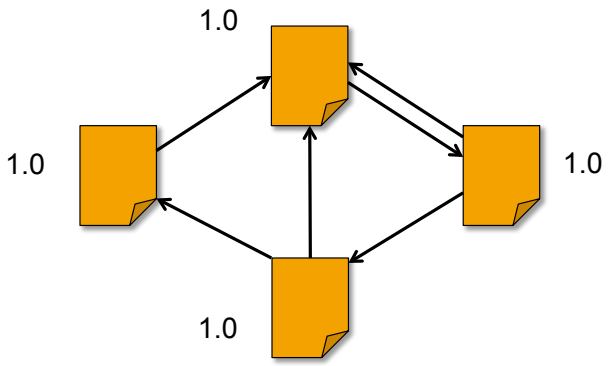
PageRank: An Example (cont.)

$$R(u) = d \sum_{v \in B_u} \frac{R(v)}{N_v} = d(R(v_1) * \frac{1}{N_{v_1}} + R(v_2) * \frac{1}{N_{v_2}} + \ldots + R(v_k)\frac{1}{N_{v_k}})$$

- To compute page rank for page u, we sum the weighted PageRanks of all pages v_i (i=1,2…k), multiplied with a damping factor d which can be set between 0 and 1.
- By using the damping factor, the extend of PageRank benefit for a page by another page linking to it is reduced by a factor d.

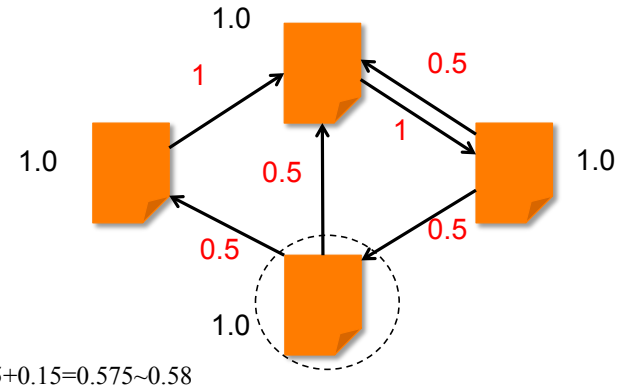
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute certain portion of its current rank ($rank_p$ / | $neighbors_p$ |) to its (outgoing) neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs (d=0.85)



Algorithm

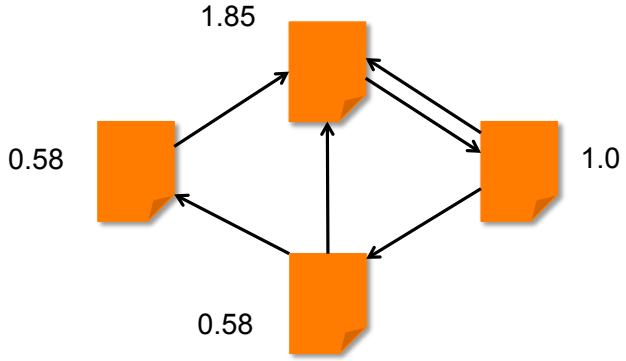
1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / | $neighbors_p$ | to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



0.50*0.85+0.15=0.425+0.15=0.575~0.58

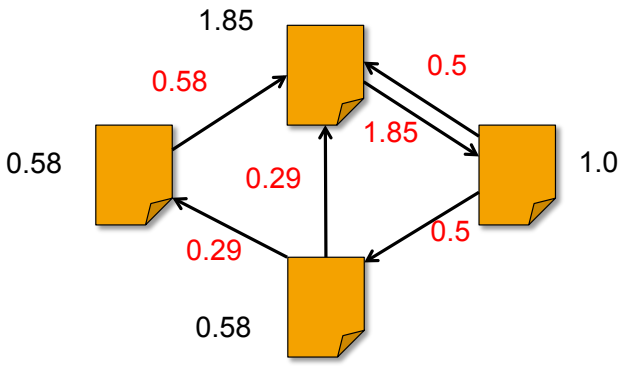
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / | $neighbors_p$ | to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



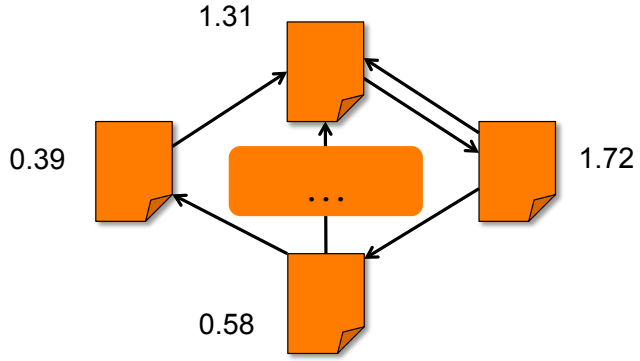
Algorithm (2nd iteration)

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / | $neighbors_p$ | to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



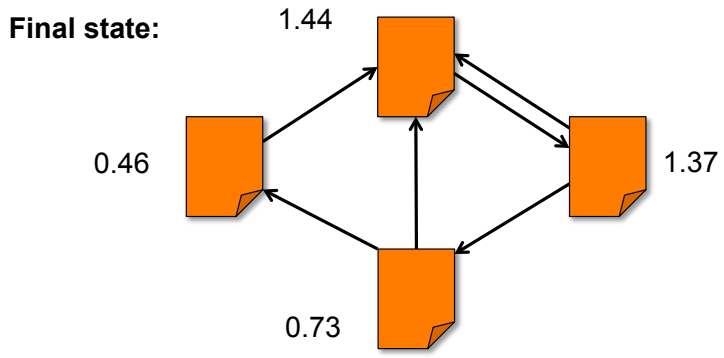
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / | $neighbors_p$ | to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs

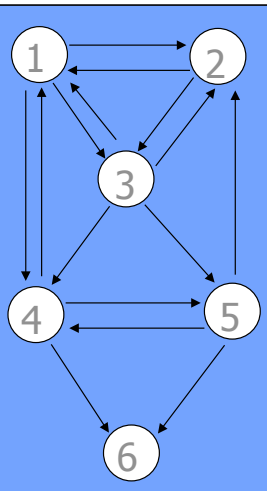


Algorithm (final state)

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / | $neighbors_p$ | to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



Transition Matrix Example (6-node sample web)



- Transition matrix:

$$T = \begin{bmatrix} 0 & 1/2 & 1/4 & 1/3 & 0 & 0 \\ 1/3 & 0 & 1/4 & 0 & 1/3 & 0 \\ 1/3 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/4 & 0 & 1/3 & 0 \\ 0 & 0 & 1/4 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 \end{bmatrix}$$

Transition Matrix Example (6-node sample web)

To eliminate dangling nodes and obtain a stochastic matrix, replace a column of zeros with a column of $1/n$'s, where n is the number of web pages.

$$T = \begin{bmatrix} 0 & 1/2 & 1/4 & 1/3 & 0 & 0 \\ 1/3 & 0 & 1/4 & 0 & 1/3 & 0 \\ 1/3 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/4 & 0 & 1/3 & 0 \\ 0 & 0 & 1/4 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 \end{bmatrix} \quad \bar{T} = \begin{bmatrix} 0 & 1/2 & 1/4 & 1/3 & 0 & 1/6 \\ 1/3 & 0 & 1/4 & 0 & 1/3 & 1/6 \\ 1/3 & 1/2 & 0 & 0 & 0 & 1/6 \\ 1/3 & 0 & 1/4 & 0 & 1/3 & 1/6 \\ 0 & 0 & 1/4 & 1/3 & 0 & 1/6 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/6 \end{bmatrix}$$

What's wrong with the strawman implementation?

- No consideration on memory capacity
- No consideration on Performance
  - how to store the vertex set and the edge set to make the access to vertices and edges fast (super-efficient)? → index, cache …
- Concurrent Access and Consistency
- Recovery in the presence of hardware or software failure
- …

What's wrong with the strawman implementation?

- Matrix representation can fit into the memory
  - Access graph in row by row layout in memory and on disk: no flexibility to scale
  - Strong Assumption: one row must fit into the memory at the minimum
  - Algorithm execution incur out of memory (OOM) error
    - when a high degree vertex and its adjacency list plus intermediate result cannot fit into the available working memory

What would you do

if the graph and its intermediate computation results cannot fit into the available memory ?

External Memory Solutions

- Remote Memory
  - Key Innovations
    - Data Partitioning
    - Communication/ Messaging cost/ optimization
- Secondary Storage
  - Key Innovations
    - Data Placement
    - Sequential v.s. Random Access
    - Parallel processing

Execution Time: Apach Hama v.s. GraphMap

	Dataset	total execution time (sec)					
		SSSP		CC		PageRank	
		Hama	Graph Map	Hama	Graph Map	Hama	Graph Map
	hollywood-2011	108.776	18.347	177.854	39.365	268.474	111.466
	orkut	108.744	21.345	195.841	54.383	286.054	111.46
	cit-Patents	27.693	12.337	24.646	12.335	30.688	18.353
	soc-LiveJournal1	48.697	18.346	60.734	33.357	75.76	39.369
8.6GB	uk-2005	Fail	156.49	Fail	706.329	Fail	573.964
24GB	twitter	Fail	150.486	Fail	303.653	Fail	1492.966

12GB DRAM per node of a cluster of size 21 nodes compared to 252GB distributed shared memory

Analysis

- Hama fails for large graphs with more than 900M edges while GraphMap still works
- Note that, in all the cases, GraphMap is faster (up to 6 times) than Apach Hama

Lee,Liu+, IEEE SuperComputer 2015

What's wrong with the simple strawman implementation?

- Lack of consideration on performance optimization
  - Skewed vertex degree distribution
  - Skewed edge weight distribution
  - …

Real-world Big Graphs (Highly skewed)

Graph	Type	#Vertices	#Edges	AvgDeg	MaxIn	MaxOut
Yahoo	directed	1.4B	6.6B	4.7	7.6M	2.5K
uk-union	directed	133.6M	5.5B	41.22	6.4M	22.4K
uk-2007-05	directed	105.9M	3.7B	35.31	975.4K	15.4K
Twitter	directed	41.7M	1.5B	35.25	770.1K	3.0M
Facebook	undirected	5.2M	47.2M	18.04	1.1K	1.1K
DBLPS	undirected	1.3M	32.0M	40.67	1.7K	1.7K
DBLPM	undirected	0.96M	10.1M	21.12	1.0K	1.0K
Last.fm	undirected	2.5M	42.8M	34.23	33.2K	33.2K

Multi-thread Parallel Processing: Three Known Problems

- Low CPU utilization and High Memory bandwidth
  - Vertex state is small but vertex degree is high and incur high memory bandwidth to access the whole adjacency list of a high degree vertex
- Global Synchronization (Critical Section)
  - Iterative graph computation requires global synchronization in each iteration round
- Lagger thread
  - thread handling high degree vertices compared to thread handling low degree vertices

What would you do

if the graph is highly skewed ?

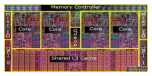
Skewedness: Novel Data Structures

- Vertex Degree Skewedness
- Vertices have skewed neighbors (both in-edge and out-edge)
- Edge Weight Skewedness
- Some edges have tiny weights and others significant weights

Parallel Graph Processing @ DiSL, Georgia Tech

- Developing Multi-tier Graph Parallel Abstractions
  - Top Tier: Subgraph centric
  - Bottom Tier: : Vertex centric or Edge centric
- **NEW FRAMEWORKS for Parallel Graph Processing**
  - Single Machine Solutions
    - GraphLego [ACM HPDC 2015] → Parallel Abstractions
    - GraphTwist [VLDB2015] → Two Tier Optimizations
  - Distributed Approaches
    - GraphMap [IEEE SC 2015]
    - PathGraph [IEEE SC 2014], joint with Dr. Pingpeng Yuan et.al @HUST

GPUs

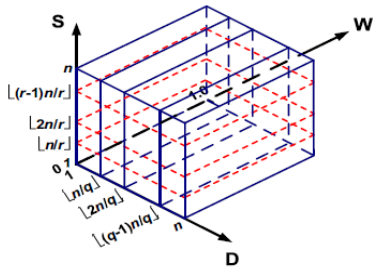

Multicore


Clusters

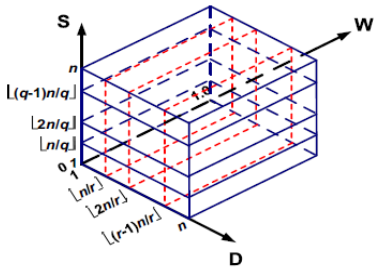

Science Clouds

Commercial Clouds

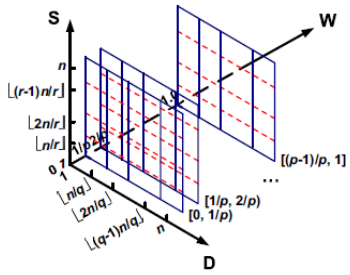
Multi-level Hierarchical Graph Parallel Abstractions [Zhou,Liu VLDB2015]



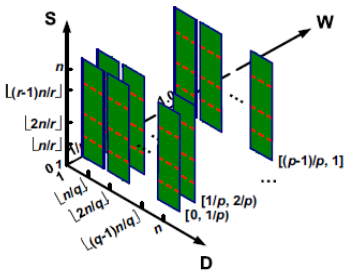
(a) In-edge Cube



(b) Out-edge Cube

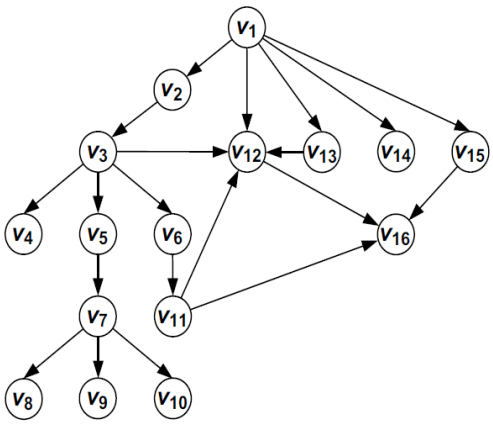


(c) In-edge Slice

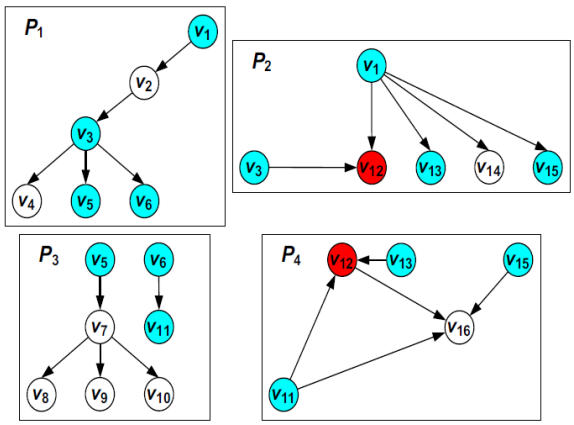


(d) In-edge Strip

Dice Partitioning: An Example



(a) Original Graph



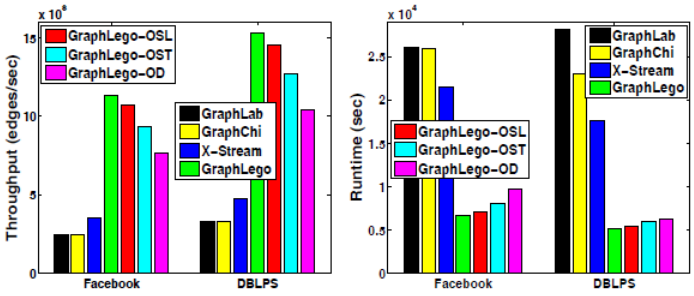
(b) Dice Partitioning

, v5,v6,v7, v11, v12, v15

DVP: v2,v3,v4,v5,v6, v7,v8,v9,v10,v11, v12,v13,v14,v15.v16

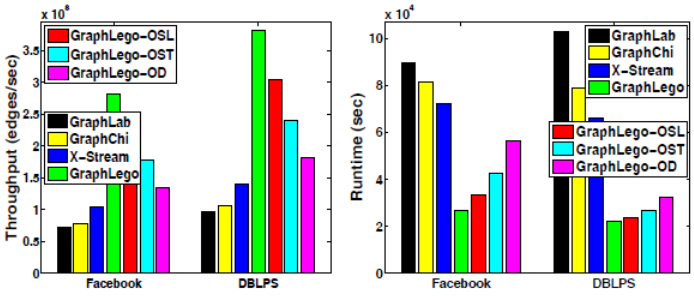
[Zhou,Liu VLDB2015]

Execution Efficiency on Multiple Graphs



(a) Throughput (b) Runtime

Diffusion Kernel on Two Real Graphs



(a) Throughput (b) Runtime

Inc-Cluster on Two Real Graphs [Zhou,Liu VLDB2015]

What's wrong with the simple strawman implementation?

- Lack of graph abstractions that promote Parallel Computation
  - Some vertex has very short computation time in each iteration and other vertex has relatively long computation time
  - CPU time is very short compared to data access time
  - …

How would you add more parallel processing
if you need to speed up the graph analysis algorithm ?

Parallel Graph Abstractions

- Distributed graph processing
  - How to distribute a graph among a cluster of servers?
    - Increase parallel processing performance
    - Minimize inter-server Messaging cost
  - Synchronization among iterations of the algorithm
- Single server solution
  - Mismatch between CPU and memory
  - Imbalance among Parallel vertex processing tasks
  - Synchronization among iterations of the algorithm

What's wrong with the simple strawman implementation?

- No system support for failure recovery

How would you handle failure when the graph analysis task is a long running (multi-round iterative computation) job ?

What's wrong with the simple strawman implementation?

- Systems that are designed and optimized for iterative graph computation
  - Cannot be used effectively for other graph manipulations
    - Search expensive; no indexes; always read entire graph
      - e.g., Cannot find subgraph with given features quickly
    - Brute force query processing for graph queries (subgraph matching)
      - Did we need to look at all vertices and edges of the connected components of the graph?
    - . . .

Other Complications:

- Small Graphs v.s. Big Graphs
  - Small graph: can fit whole graph and its processing in main memory
  - Big Graph: cannot fit the whole graph and its processing in main memory
- Simple Graphs v.s. Multigraphs
  - **Simple graph**: allow only one edge per pair of vertices
  - **Multigraph**: allow multiple parallel edges between pairs of vertices
- Singe Graph Computation and Multiplication of Graphs
- …

What's wrong with the simple strawman implementation?

- Conventional cache is ineffective
- No good buffer manager: Everything comes off the disk all the time

e.g., Need graph-specific caching

What's wrong with the simple strawman implementation?

- No concurrency control
  - Single user runtime, one job on a big graph dataset each time
  - Open problem: How can multiple users access / modify a big graph file at the same time consistently

What's wrong with the simple strawman implementation?

- No reliability

e.g., - Can lose data in a crash
expensive, start multi-iteration job from scratch today!
- Can leave operations half done
how to recover in the presence of failure?

What's wrong with the simple strawman implementation?

- No security

e.g., - File system insecure
- File system protection too coarse
- Graph data sensitivity, graph computation output sensitivity, etc.

What's wrong with the simple strawman implementation?

- No flexible and easy to use application program interface (API)

e.g., how can a payroll program get the data

Course Overview

- Big Data Systems
  - Implementation Algorithms
  - Optimization opportunities and techniques
  - Experimentation Methods and algorithms
- Data Analytics
  - Fundamental of classic data mining/machine learning algorithms
  - Innovation demands in Deep learning for different types of big data

Big Data Systems

- Research and Development Techniques and Challenges
  - Data Storage and Data Structures
    - PCM, NVRAM, FLASH, etc.
    - Data Utility driven optimizations
  - Basic and Complex Data Manipulation and Performance Optimization
  - Parallel Abstractions and Data driven Parallelization
  - Crash Recovery and Concurrency Control
  - Security & Privacy, Trust and Integrity

    Authorization, encryption, authentication …
  - Distributed Big Data Systems
    - Data distribution and placement, Computation Partitioning and distribution, Synchronization and interoperation, distributed recovery, …

Data Analytics

- Research and Development Trends and Big Data Challenges
  - Descriptive Learning Models
  - Predictive Learning Models
  - Supervised Learning (Classical ML) and Unsupervised Learning (Classical DM)
  - Text mining, Spatial mining, Mobile trajectory mining, Graph mining, Time Series mining, Multimedia mining, etc.
  - Domain specific data analytics problems in Business, Science and Engineering disciplines
  - Ensemble learning models and algorithms

Summary:

- Today's Lecture
  - Syllabus
  - Why implementation of big data systems is not simple
- Homework Assignment: Questionnaire

Questions?