Arachne: An Open-Source Framework for Interactive Massive-Scale Graph Analytics

David A. Bader

@Prof_DavidBader

http://www.cs.njit.edu/~bader

NJIT
New Jersey Institute of Technology

David A. Bader

Distinguished Professor and
Director, Institute for Data Science

- IEEE Fellow, ACM Fellow, SIAM Fellow, AAAS Fellow
- IEEE Sidney Fernbach Award
- 2022 inductee into University of Maryland's Innovation Hall of Fame, A. James Clark School of Engineering
- Recent Service:
  - White House's National Strategic Computing Initiative (NSCI) panel
  - Computing Research Association Board
  - Chair, NSF Committee of Visitors for Office of Advanced Cyberinfrastructure
  - NSF Advisory Committee on Cyberinfrastructure
  - Council on Competitiveness HPC Advisory Committee
  - IEEE Computer Society Board of Governors
  - IEEE IPDPS Steering Committee
  - Editor-in-Chief, ACM Transactions on Parallel Computing
  - Editor-in-Chief, IEEE Transactions on Parallel and Distributed Systems
- Over $188M of research awards
- 300+ publications, ≥ 13,900 citations, h-index ≥ 66
- National Science Foundation CAREER Award recipient
- Directed: Facebook AI Systems
- Directed: NVIDIA GPU Center of Excellence, NVIDIA AI Lab (NVAIL)
- Directed: Sony-Toshiba-IBM Center for the Cell/B.E. Processor
- Founder: Graph500 List benchmarking "Big Data" platforms
- Recognized as a "RockStar" of High Performance Computing by InsideHPC in 2012 and as HPCwire's People to Watch in 2012 and 2014.

New Jersey Institute of Technology

Today’s talk is dedicated to my uncle

Elliot Norman Ashrey

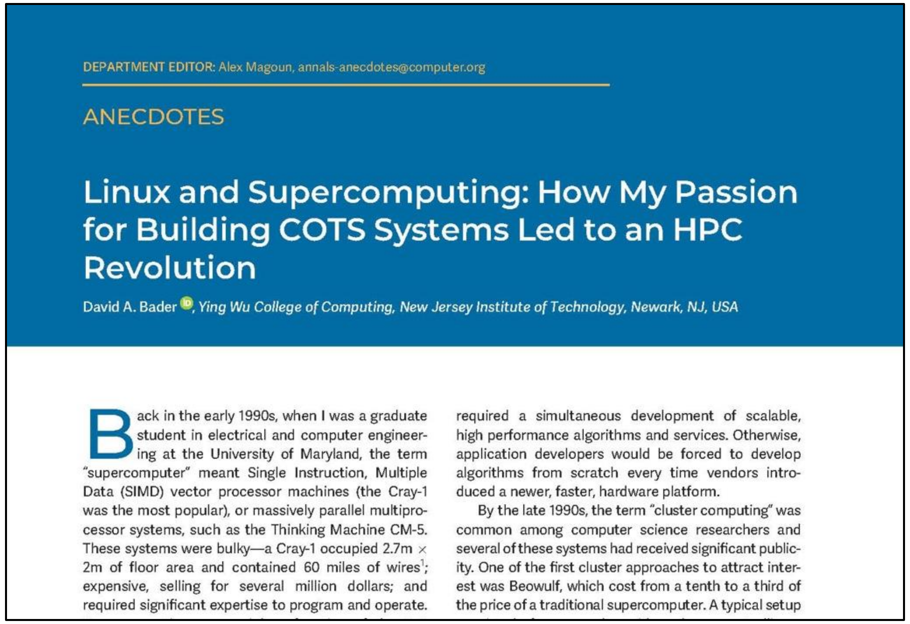
- B: 23 Sep 1931 in New York City, New York, USA
- D: 26 Apr 2024 in New York City, New York, USA

2021 IEEE Sidney Fernbach Award

David Bader cited for the development of Linux-based massively parallel production computers and for pioneering contributions to scalable discrete parallel algorithms for real-world applications.

2022 IEEE Computer Society President Bill Gropp presents David Bader with the Sidney Fernbach Award at SC21

1998: Bader Invents the Linux Supercomputer



DEPARTMENT EDITOR: Alex Magoun, annals-anecdotes@computer.org

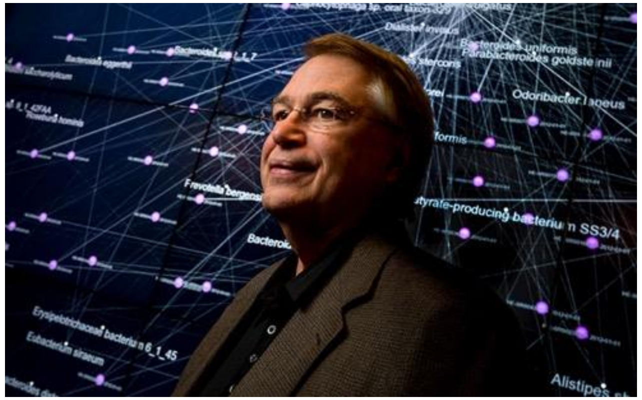
ANECDOTES

Linux and Supercomputing: How My Passion for Building COTS Systems Led to an HPC Revolution

David A. Bader, Ying Wu College of Computing, New Jersey Institute of Technology, Newark, NJ, USA

Back in the early 1990s, when I was a graduate student in electrical and computer engineering at the University of Maryland, the term "supercomputer" meant Single Instruction, Multiple Data (SIMD) vector processor machines (the Cray-1 was the most popular), or massively parallel multiprocessor systems, such as the Thinking Machine CM-5. These systems were bulky—a Cray-1 occupied 2.7m × 2m of floor area and contained 60 miles of wires[1]; expensive, selling for several million dollars; and required significant expertise to program and operate. required a simultaneous development of scalable, high performance algorithms and services. Otherwise, application developers would be forced to develop algorithms from scratch every time vendors introduced a newer, faster, hardware platform.

By the late 1990s, the term "cluster computing" was common among computer science researchers and several of these systems had received significant publicity. One of the first cluster approaches to attract interest was Beowulf, which cost from a tenth to a third of the price of a traditional supercomputer. A typical setup



"This effort of yours has enormous historic resonance,"
– Larry Smarr, Distinguished Professor Emeritus, UC San Diego
Founding Director of NCSA, Founding Director of Calit2

Source: UC San Diego
https://ucsdnews.ucsd.edu/pressrelease/pioneering-scientist-and-innovator-larry-smarr-retires

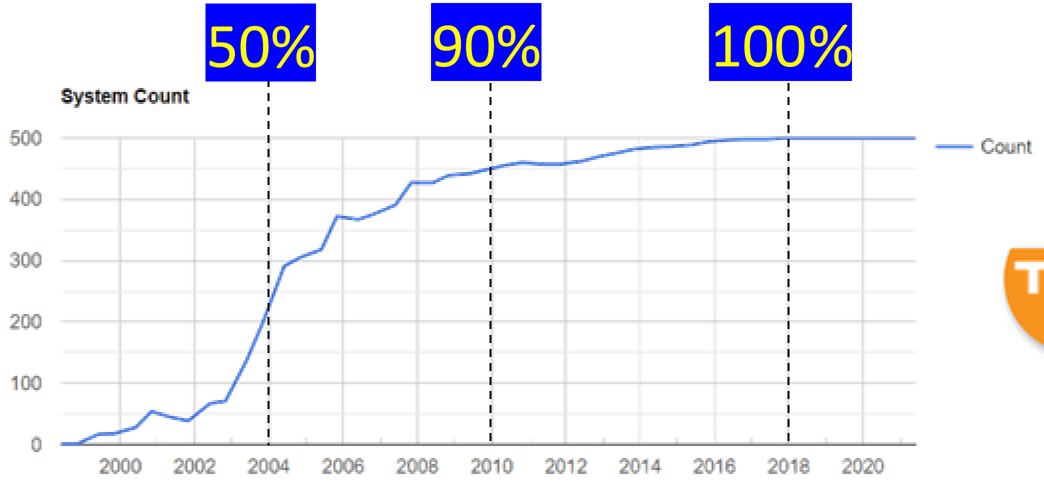
NJIT New Jersey Institute of Technology

Impact: Top500 Supercomputers Running Linux

Photo credit: Information Week, 2008

"Today, 100% of the Top 500 supercomputers in the world are Linux HPC systems, based on Bader's technical contributions and leadership. This is one of the most significant technical foundations of HPC."

– Steve Wallach is a guest scientist for Los Alamos National Laboratory and 2008 IEEE CS Seymour Cray Computer Engineering Award recipient.



TOP500 SUPERCOMPUTER SITES

Source: http://www.Top500.org/

NJIT New Jersey Institute of Technology

NJIT

New Jersey Institute of Technology

New Jersey Institute of Technology

“NJIT Climbs the Rankings of U.S. News & World Report, A Top 50 Public University” – 13 Sep 2021

“NJIT Named As One of Nation's 'Best Colleges' for 2022, The Princeton Review Says” – 6 Sep 2021

“Wall Street Journal/College Pulse Ranks NJIT No. 2 Public University in the US” – 6 Sep 2023

NJIT
New Jersey Institute of Technology

Launched in **July 2019**, with inaugural director
David A. Badder

(~40 faculty in current centers)

Solving real-world challenges

- Urban sustainability
- Healthcare analytics
- Trustworthy, Free and Fair Elections
- Insider threat detection
- Utility infrastructure protection
- Cyberattack defense
- Disease outbreak and epidemic monitoring

Center for Big Data
- Big Data Analytics, Systems, and Tools
- Cyberinfrastructure

Cybersecurity Research Center
- Practical encryption
- Privacy technologies
- Information Assurance

The Structural Analysis of Biomedical Ontologies Center
- Medical Informatics
- NIH / National Cancer Institute

FinTech Group
- Financial Services
- Insurance Industry

Machine Learning & AI
- Real-world technologies
- Industrial partnerships

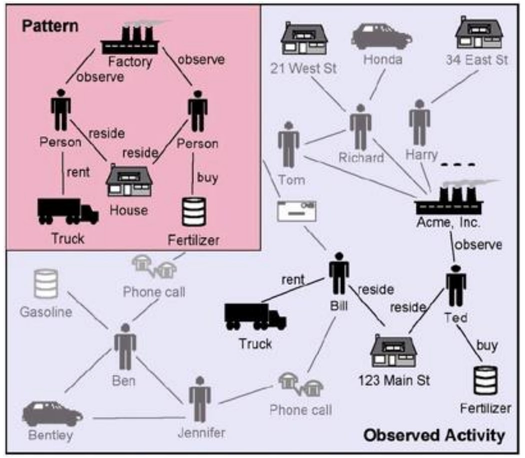
NJIT
New Jersey Institute of Technology

Data-Quad

Unknown

Known **Objects** Unknown

Patterns



Known

NJIT New Jersey Institute of Technology

Graph Data Science: Real-world challenges

All involve exascale streaming graphs:

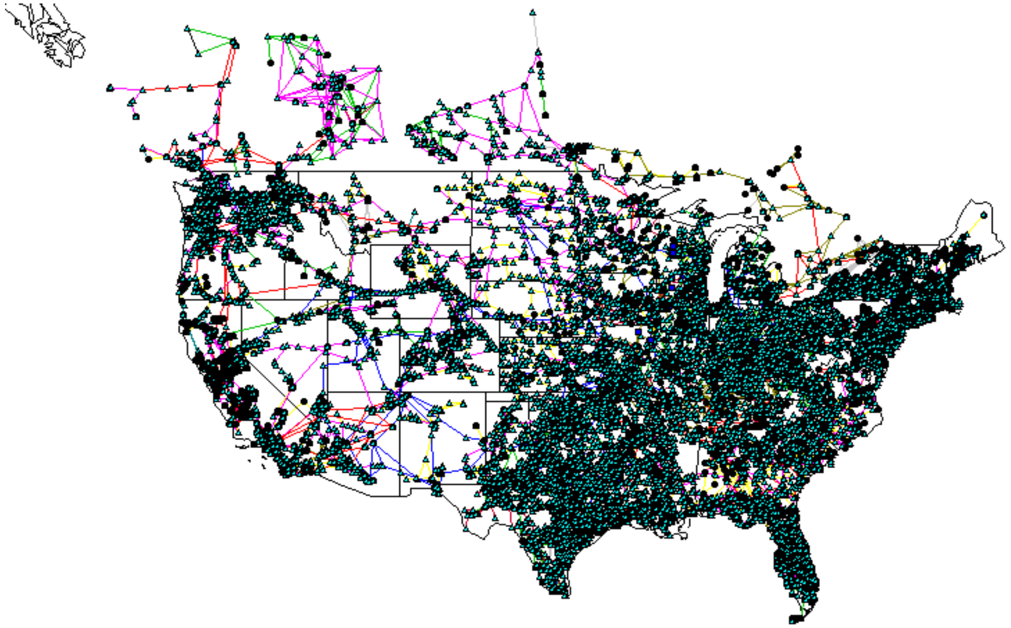
- **Health care →** disease spread, detection and prevention of epidemics/pandemics (e.g. SARS, Avian flu, H1N1 “swine” flu)
- **Massive social networks →** understanding communities, intentions, population dynamics, pandemic spread, transportation and evacuation
- **Intelligence →** business analytics, anomaly detection, security, knowledge discovery from massive data sets
- **Systems Biology →** understanding complex life systems, drug design, microbial research, unravel the mysteries of the HIV virus; understand life, disease,
- **Electric Power Grid →** communication, transportation, energy, water, food supply
- **Modeling and Simulation →** Perform full-scale economic-social-political simulations

REQUIRES PREDICTING / INFLUENCE CHANGE IN REAL-TIME AT SCALE

NJIT New Jersey Institute of Technology

Massive Data Analytics: Infrastructure

- The U.S. high-voltage transmission grid has >150,000 miles of line.
- Real-time detection of changes and anomalies in the grid is a large-scale problem.
- May mitigate impact of widespread blackouts due to equipment failure or intentional damage.



The New York Times
Thursday, September 4, 2008

Report on Blackout Is Said To Describe Failure to React

By MATTHEW L. WALD
Published: November 12, 2003

A report on the Aug. 14 blackout identifies specific lapses by various parties, including FirstEnergy's failure to react properly to the loss of a transmission line, people who have seen drafts of it say.

A working group of experts from eight states and Canada will meet in private on Wednesday to evaluate the report, people involved in the

E-MAIL
PRINT
SINGLE-PAGE
REPRINTS
SAVE
SHARE

New Jersey Institute of Technology

Network Analysis for Intelligence and Surveillance

- [Krebs ’04] Post 9/11 Terrorist Network Analysis from public domain information
- Plot masterminds correctly identified from interaction patterns: centrality

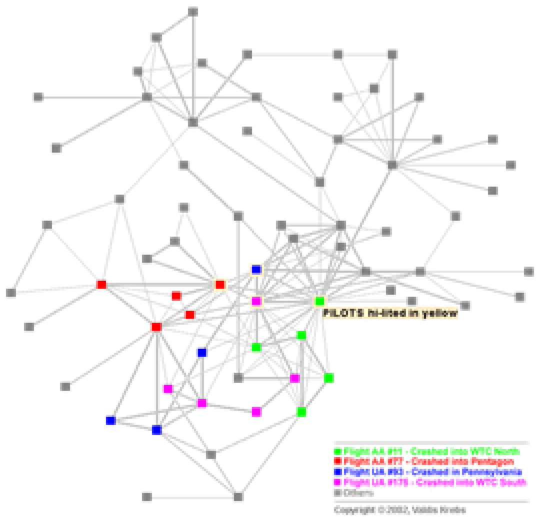


Image Source: http://www.orgnet.com/hijackers.html

- A global view of entities is often more insightful
- Detect anomalous activities by exact/approximate graph matching

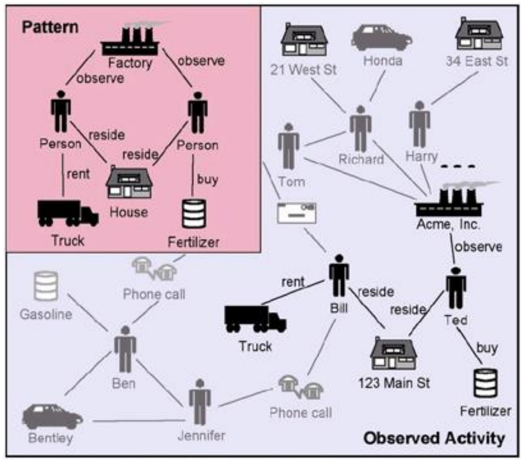
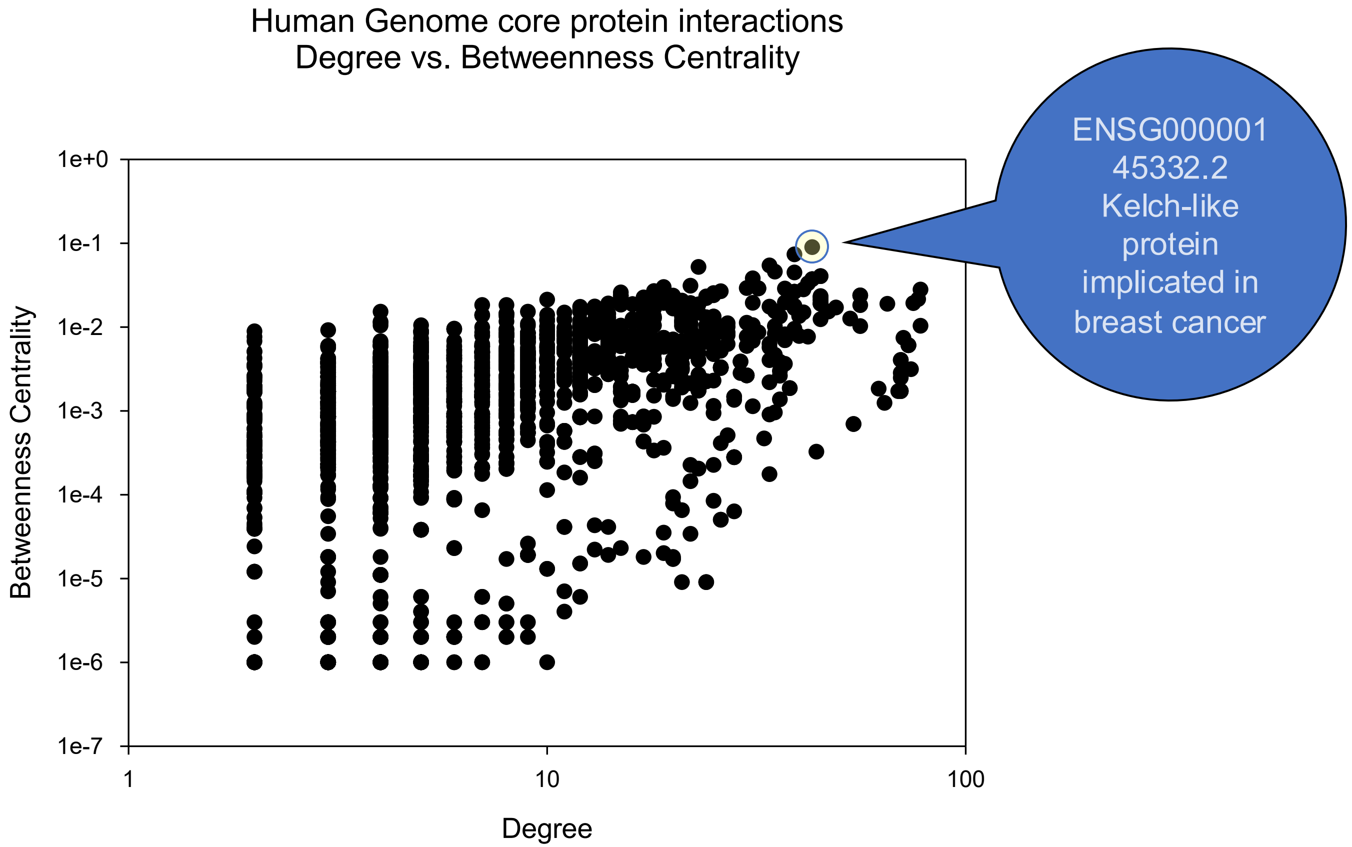


Image Source: T. Coffman, S. Greenblatt, S. Marcus, Graph-based technologies for intelligence analysis, CACM, 47 (3, March 2004): pp 45-47

Massive Data Analytics: Public Health

- CDC/national-scale surveillance of public health
- Cancer genomics and drug design
    - Computed Betweenness Centrality of Human Proteome



Mining Twitter for Social Good

ICPP 2010

Massive Social Network Analysis: Mining Twitter for Social Good

David Ediger Karl Jiang Jason Riedy David A. Bader
Georgia Institute of Technology
Atlanta, GA, USA

Courtney Corley Rob Farber
Pacific Northwest National Lab.
Richland, WA, USA

William N. Reynolds
Least Squares Software, Inc.
Albuquerque, NM, USA

Abstract—Social networks produce an enormous quantity of data. Facebook consists of over 400 million active users sharing over 5 *billion* pieces of information each month. Analyzing this vast quantity of unstructured data presents challenges for software and hardware. We present GraphCT, a *Graph* Characterization *Toolkit* for massive graphs representing social network data. On a 128-processor Cray XMT, GraphCT estimates the betweenness centrality of an artificially generated (R-MAT) 537 million vertex, 8.6 billion edge graph in 55 minutes and a real-world graph (Kwak, *et al.*) with 61.6 million vertices and 1.47 billion edges in 105 minutes. We use GraphCT to analyze public data from Twitter, a microblogging network. Twitter's message connections appear primarily tree-structured as a news dissemination system. Within the

involves over 400 million active users with an average of 120 'friendship' connections each and sharing 5 *billion* references to items each month [11].

One analysis approach treats the interactions as graphs and applies tools from graph theory, social network analysis, and scale-free networks [29]. However, the volume of data that must be processed to apply these techniques overwhelms current computational capabilities. Even well-understood analytic methodologies require advances in both hardware and software to process the growing corpus of social media.

Social media provides staggering amounts of data.

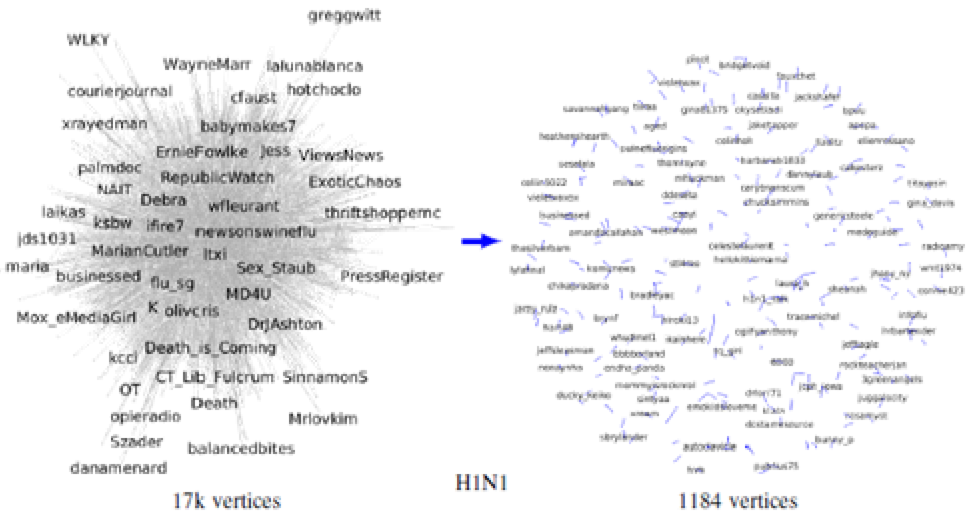


Fig. 2. Subcommunity filtering on Twitter data sets

Top 15 Users by Betweenness Centrality

Rank	Data Set H1N1	Data Set atlflood
1	@CDCFlu	@ajc
2	@addthis	@driveafastercar
3	@Official_PAX	@ATLCheap
4	@FluGov	@TWCi
5	@nytimes	@HelloNorthGA
6	@tweetmeme	@11AliveNews
7	@mercola	@WSB_TV
8	@CNN	@shaunking
9	@backstreetboys	@Carl
10	@EllieSmith_x	@SpaceyG
11	@TIME	@ATLINtownPaper
12	@CDCemergency	@TJsDJs
13	@CDC_eHealth	@ATLien
14	@perezhilton	@MarshallRamsey
15	@billmaher	@Kanye

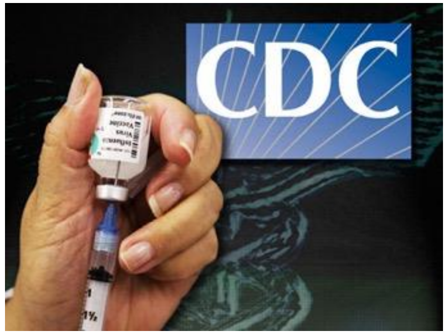


Image credit: bioethicsinstitute.org

Pacific Northwest NATIONAL LABORATORY

NJIT New Jersey Institute of Technology

Arachne:
Interactive Property Graph Analytics at Scale

Image Credit: Matias Del Carmen

New Jersey Institute of Technology

Alumni Giving MyNJIT Login
Tools
Search NJIT

Admissions Academics Research About Life at NJIT Athletics I AM A... APPLY NOW

Institute for Data Science Aims to Democratize Supercomputing With NSF Grant

Written by: Evan Koblentz

Published: Wednesday, March 17, 2021

New algorithms from at NJIT can make supercomputer power available to almost anyone

Ordinary people could soon have greater ability to analyze massive amounts of information, based on new algorithms and software tools being designed at NJIT, intended to simplify

High Performance Algorithms for Interactive Data Science at Scale
(PI: Bader)
$2.2M
3/2021 – 6/2024
NSF CCF-2109988

https://news.njit.edu/institute-data-science-aims-democratize-supercomputing-nsf-grant

New Jersey Institute of Technology

Arkouda: Dedication to **Michael H. Merrill**

(June 2, 1964 ~ November 8, 2022)

"Mike was a dedicated civil servant. He was a Computer Scientist at the Department of Defense for 34 years and was recognized in 2022 with a Distinguished Civilian Service Medal. He loved computers and technology, especially high performance computing. Mike was a problem solver and innovative thinker; he was recognized for inspiring and leading numerous large projects over the course of his career. He loved to share his knowledge and mentored many colleagues over the years—sometimes calling them his kids, sometimes his minions, but always calling them his friend."

NJIT New Jersey Institute of Technology

Productivity + Performance

python™

Chapel Programming Language

Connect with Chapel: the parallel programming language powering Arkouda

ChapelCon – free virtual event
- June 5th – Tutorial Day
- June 6th – Coding Day
- June 7th – Conference Day

Come code or chat with us!
- GitHub - https://github.com/chapel-lang/chapel
- Gitter - https://gitter.im/chapel-lang/chapel
- Discourse - https://chapel.discourse.group
- StackOverflow - https://stackoverflow.com/questions/tagged/chapel

Follow us on social media
- LinkedIn - https://www.linkedin.com/company/chapel-programming-language
- YouTube - https://www.youtube.com/@ChapelLanguage
- Twitter/X - https://x.com/ChapelLanguage
- Facebook - https://www.facebook.com/ChapelLanguage

REGISTER FOR CHAPELCON

TAKE THE CHAPEL COMMUNITY SURVEY

Arkouda + Arachne Framework

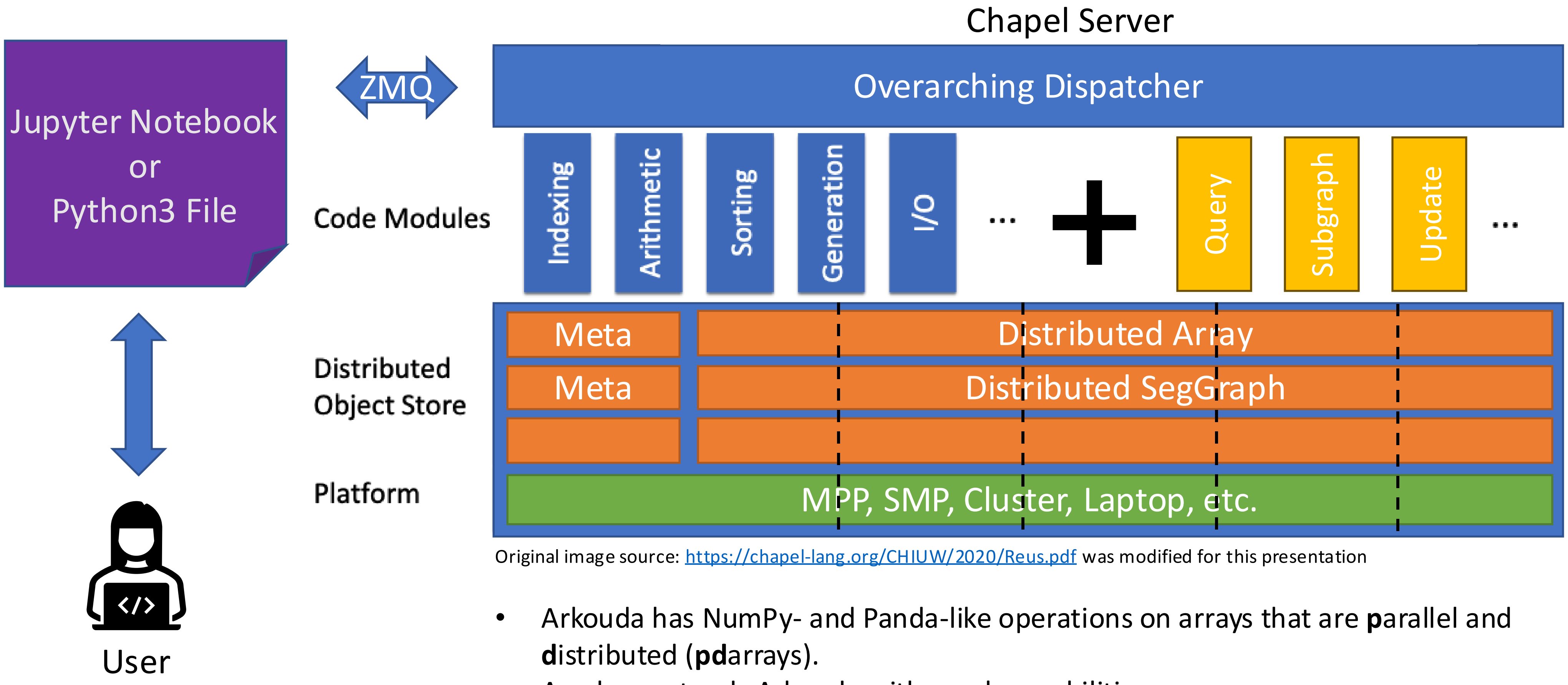
Arkouda

- an existing open-source Python framework that allows for array and dataframe operations on data that is terabytes in size but lacks graph processing operations.

Arachne

- an open-source extension to Arkouda to convert massive-scale dataframes to graphs with high-performance graph kernels and property graph capabilities while maintaining a NetworkX-like API for new Python users to easily transition to utilizing it.

Arkouda + Arachne Framework

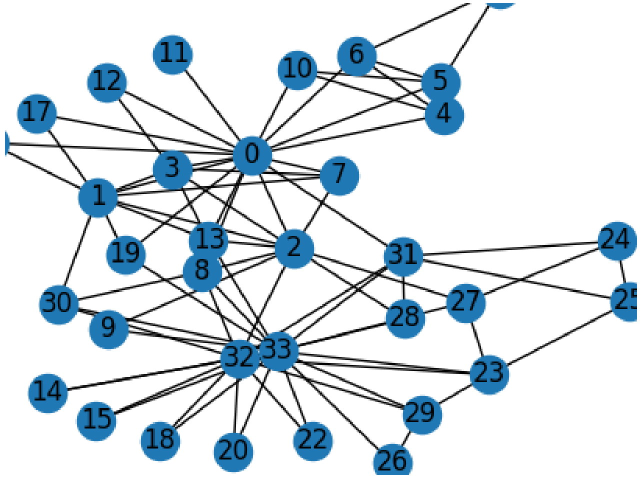


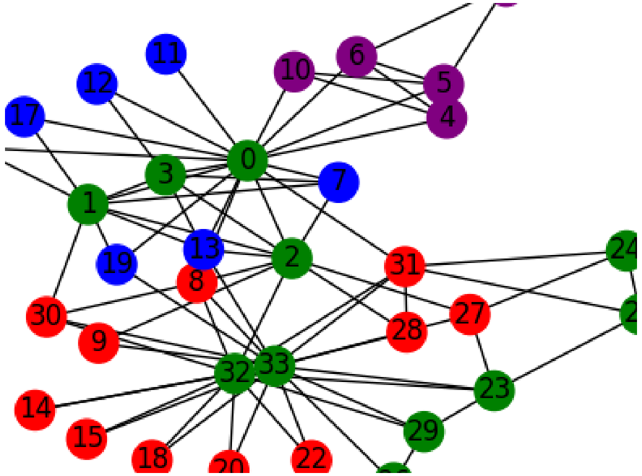
Original image source: https://chapel-lang.org/CHIUW/2020/Reus.pdf was modified for this presentation

- Arkouda has NumPy- and Panda-like operations on arrays that are **p**arallel and **d**istributed (**pd**arrays).
- Arachne extends Arkouda with graph capabilities.
- This work extends Arachne to store massive-scale graphs.
- Arachne can be thought of as a wrapper that creates a logical graph.

Karate Club Graph Example

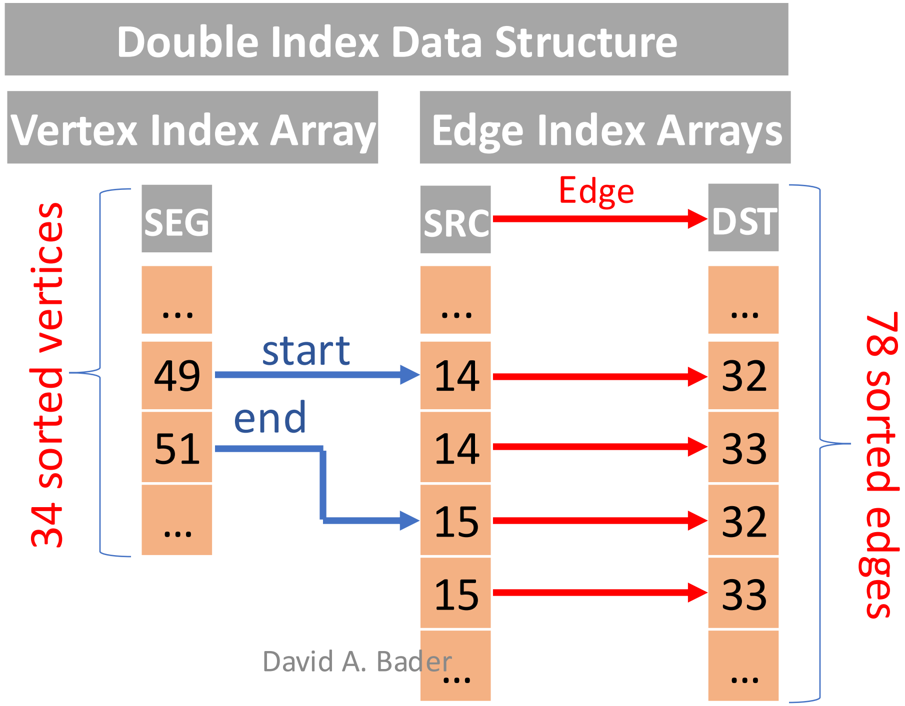
78 Edges
34 Nodes





	SRC	DST
1	0	1
…	…	…
49	14	32
50	14	33
51	15	32
52	15	33
…	…	…

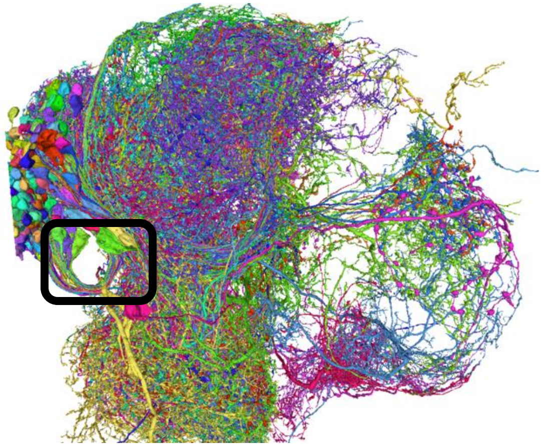
Transit data from the Arkouda data frame into an Arachne graph

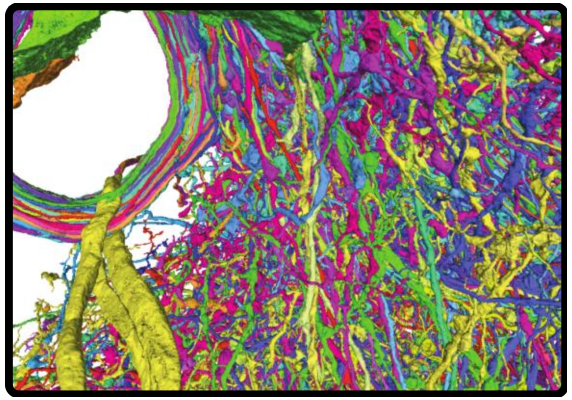


Results after running community detection using Arachne

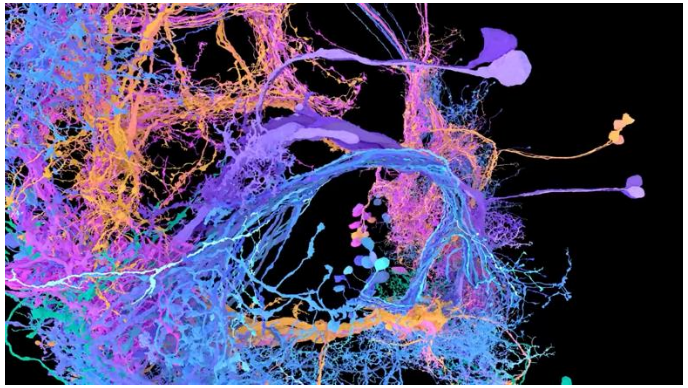
Label
0
0
…
1
1
…
0

The Connectome Project

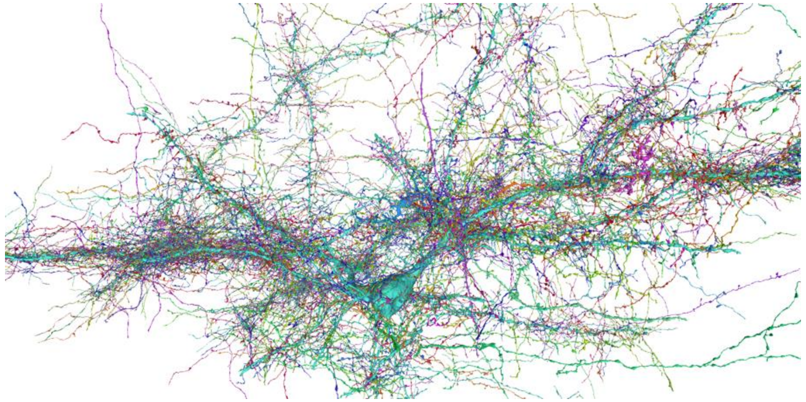




Drosophila Hemibrain Dataset, [Scheffer et al. 2020]



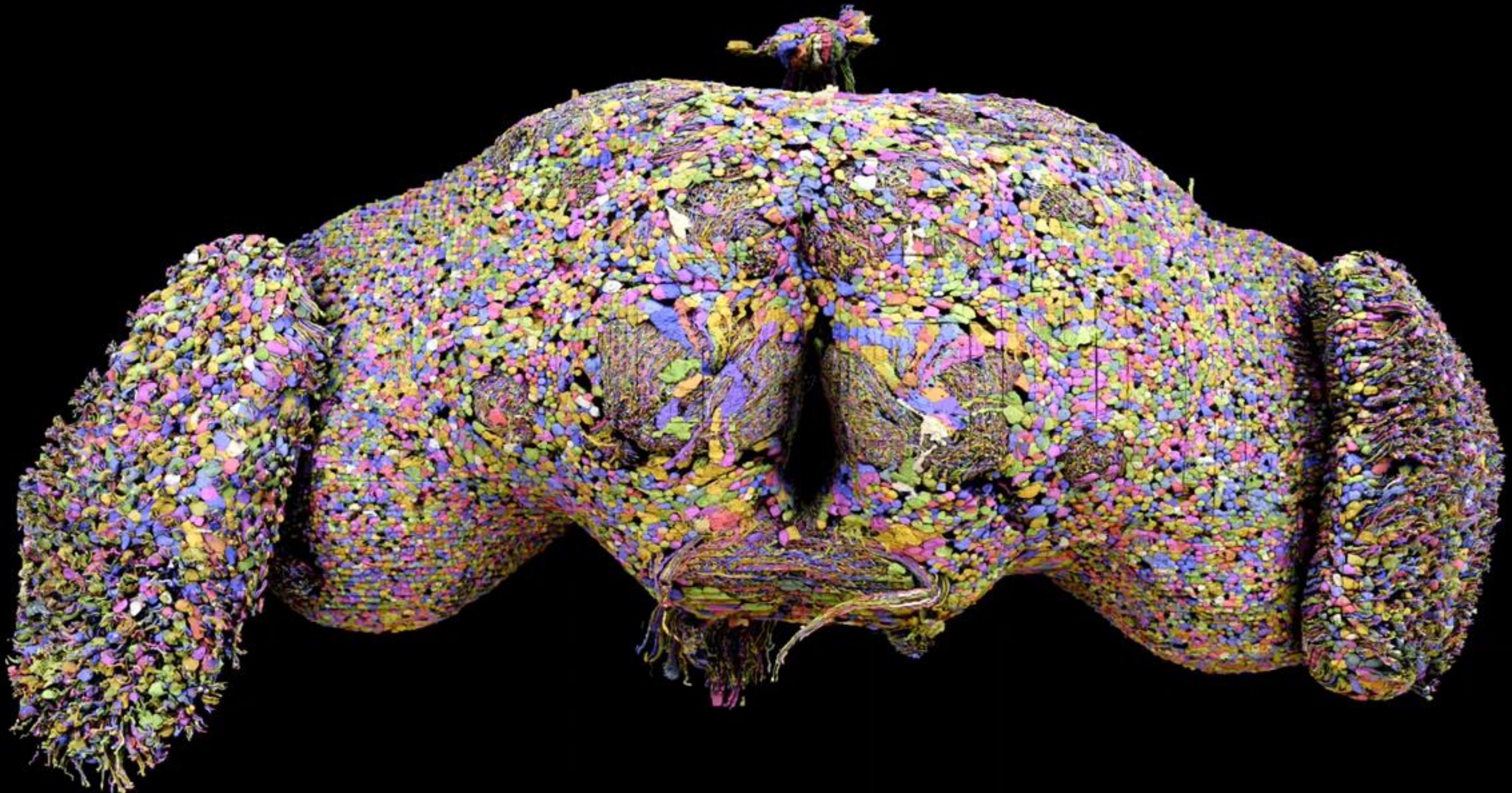
Drosophila Auditory Circuit [Baker et al. 2022]
Video: Amy Sterling, FlyWire



Slide credit: Jakob Troidl, Hanspeter Pfister, Jeff Lichtman (Harvard University)

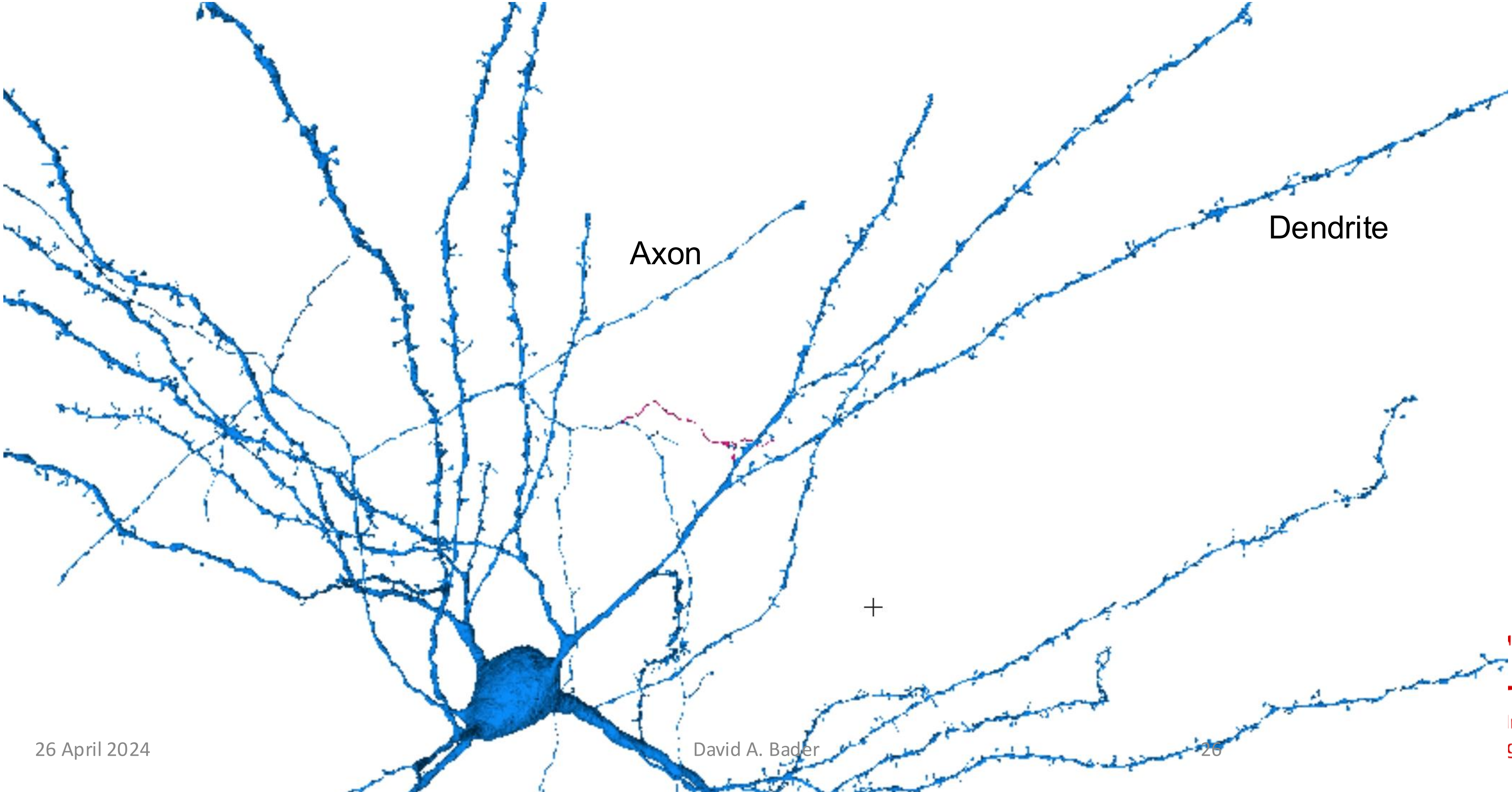
- Using Arkouda, we can covert connectome datasets with one hundred million rows of JSON objects to distributable HDF5 files in under two hours.
- Using Arachne, a graph of this size can be queried in seconds to create smaller subgraphs for deeper analysis.

NJIT New Jersey Institute of Technology

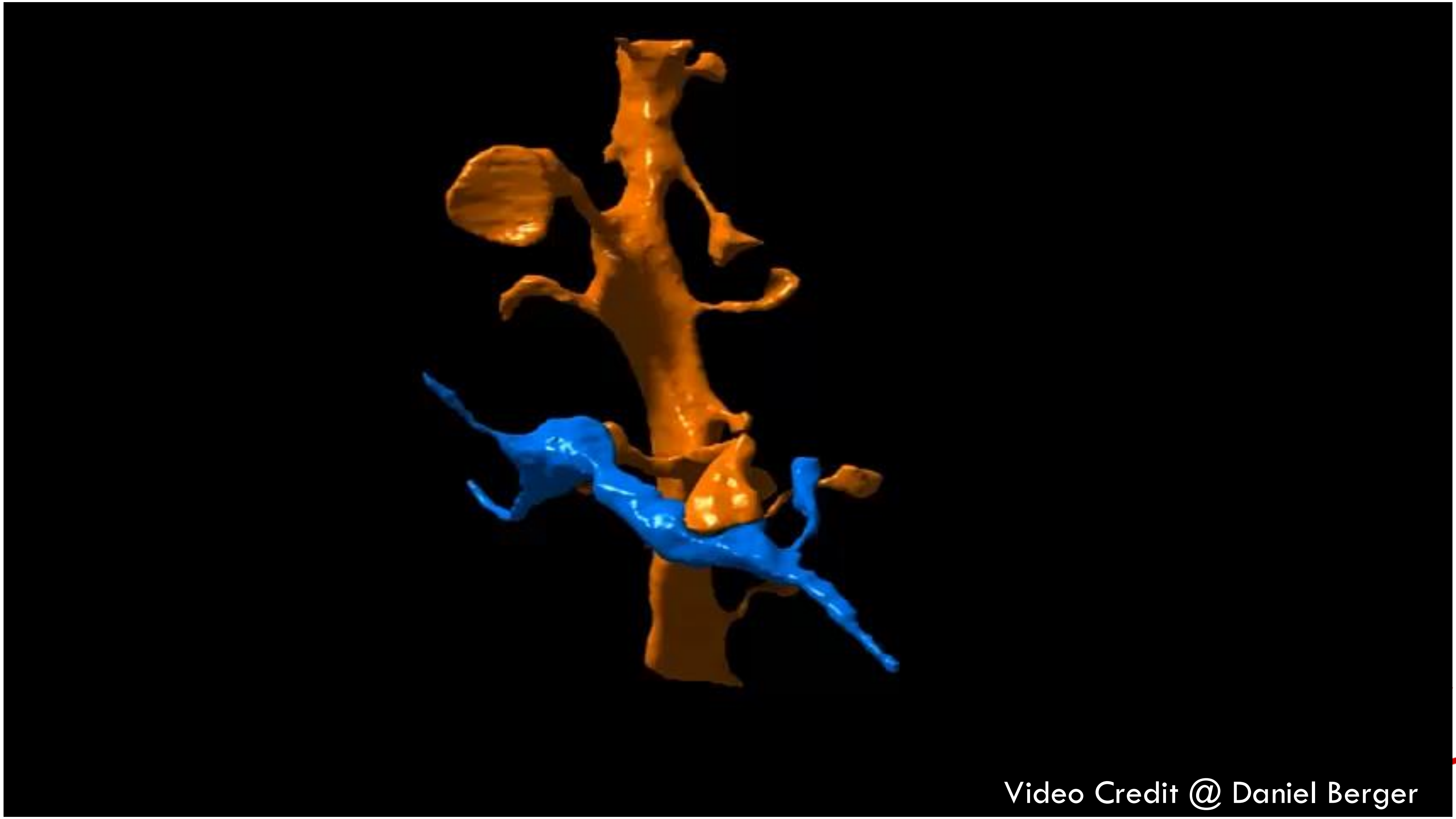


Dorkenwald et. al 2023 – bioArxiv, Animation by Tyler Sloan

Wiring Diagram of the Brain

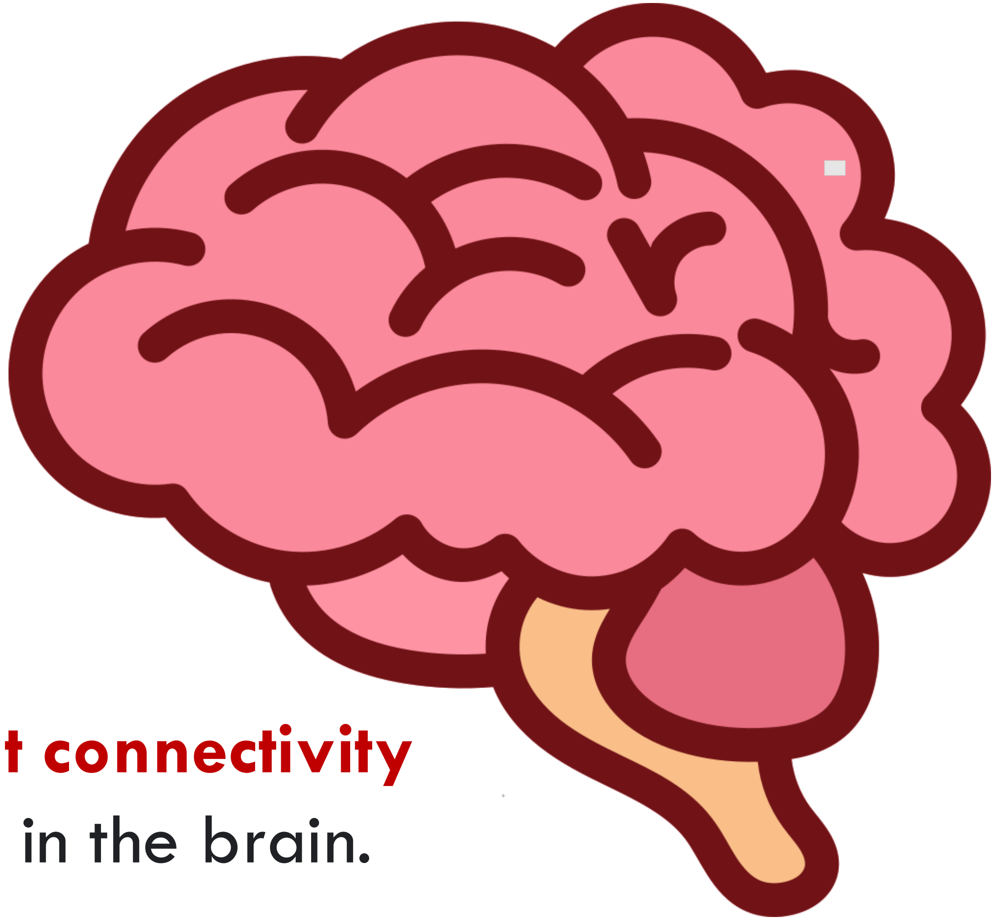


Spatial Neighborhood Analysis



New Jersey Institute of Technology

Using Graph Analytics to Understand the Brain



1 mm^3 of brain tissue

Motifs are **recurrent connectivity patterns** of neurons in the brain.

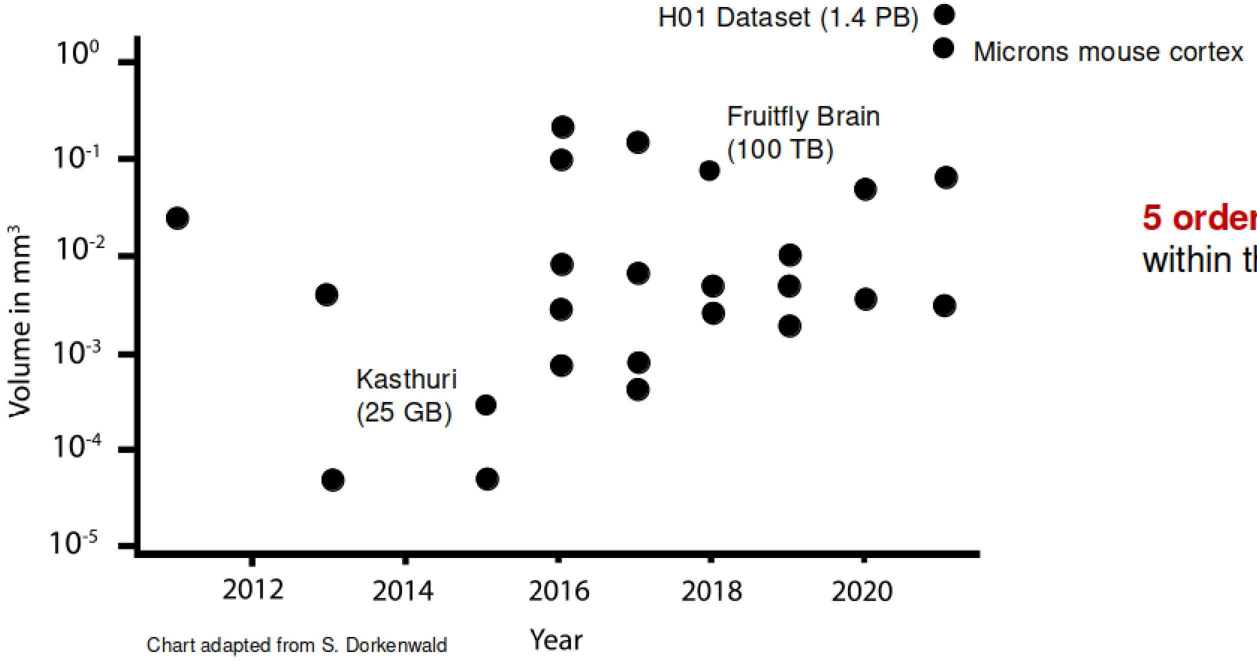
Slide credit: Jakob Troidl, Hanspeter Pfister

H01 dataset, [Shapson-Coe et al. 2021]

Lichtman Lab, Google Research

Connectome: Requires Exascale Graph Analytics

Full mouse brain
(1 Exabyte = 1 Million TB)



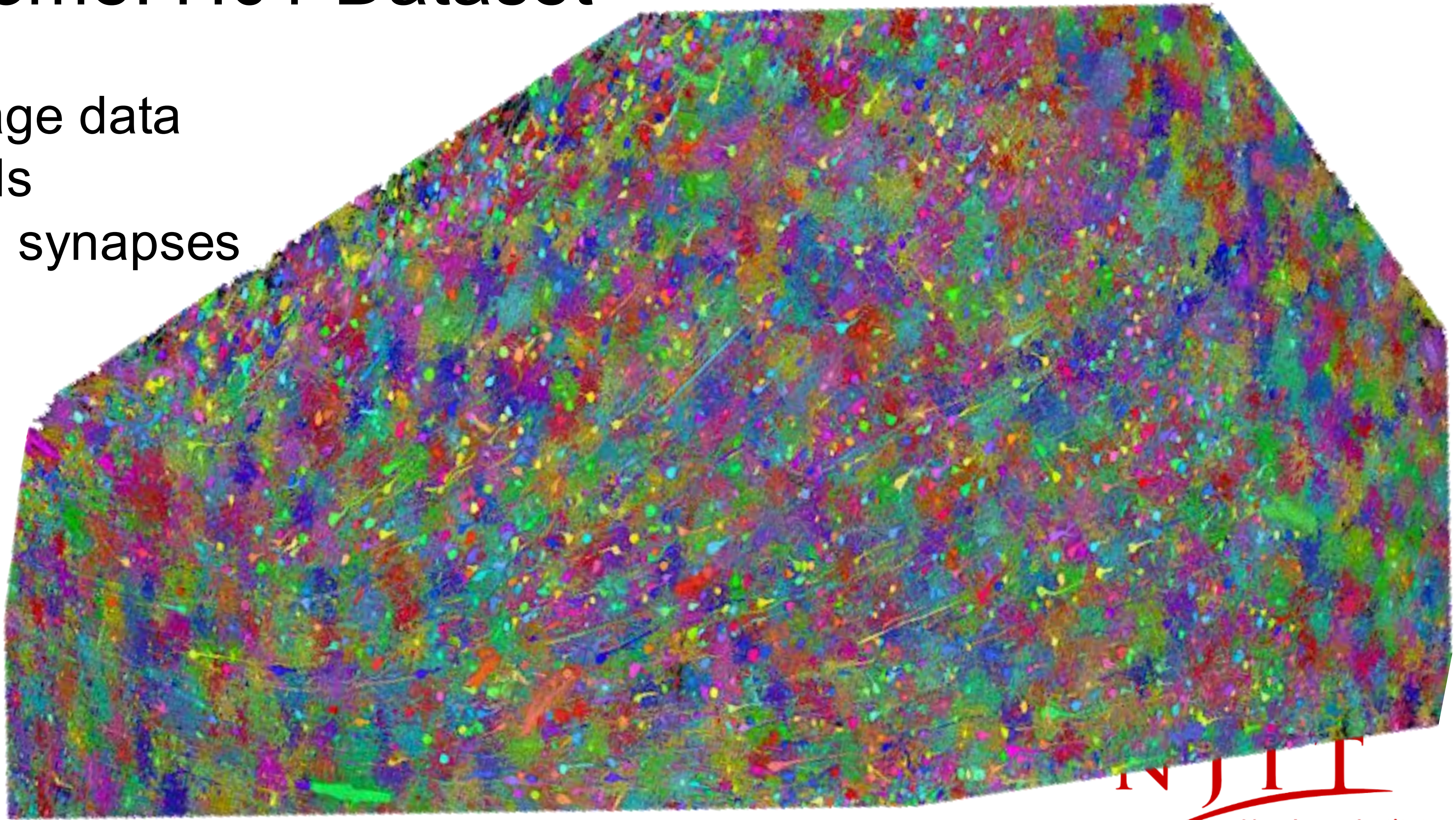
5 orders of magnitude scale up within the past 10 years

Image credit: Jakob Troidl, Hanspeter Pfister

New Jersey Institute of Technology

Connectome: H01 Dataset

~ 1.4 PB image data
~ 57,000 cells
~ 133 Million synapses



Lichtman Lab, and Google Research

Slide credit: Jakob Troidl, Hanspeter Pfister

New Jersey Institute of Technology

Connectivity motifs

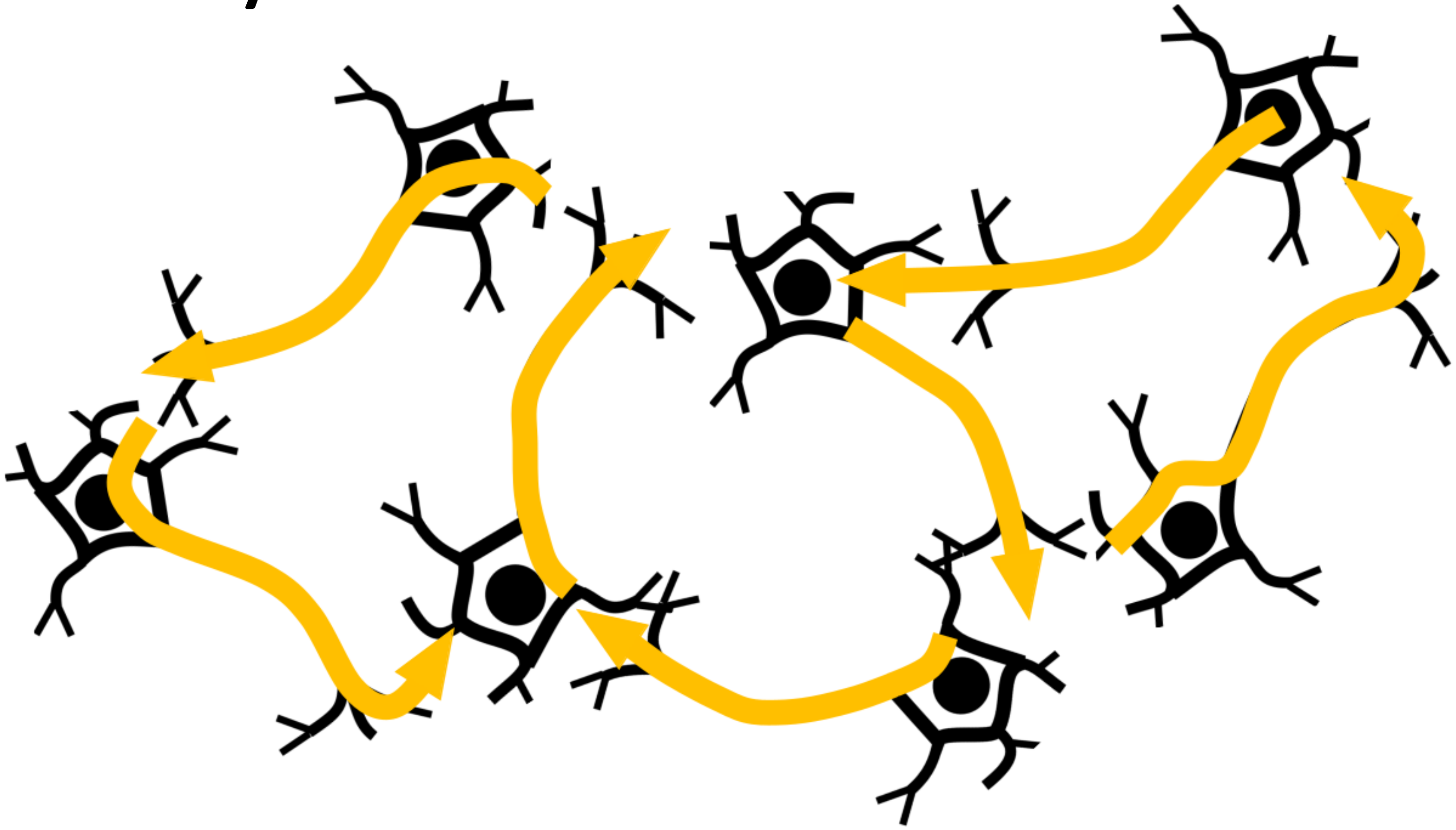


Image Credit: Wikipedia

Slide credit: Jakob Troidl, Hanspeter Pfister

New Jersey Institute of Technology

Connectivity motifs

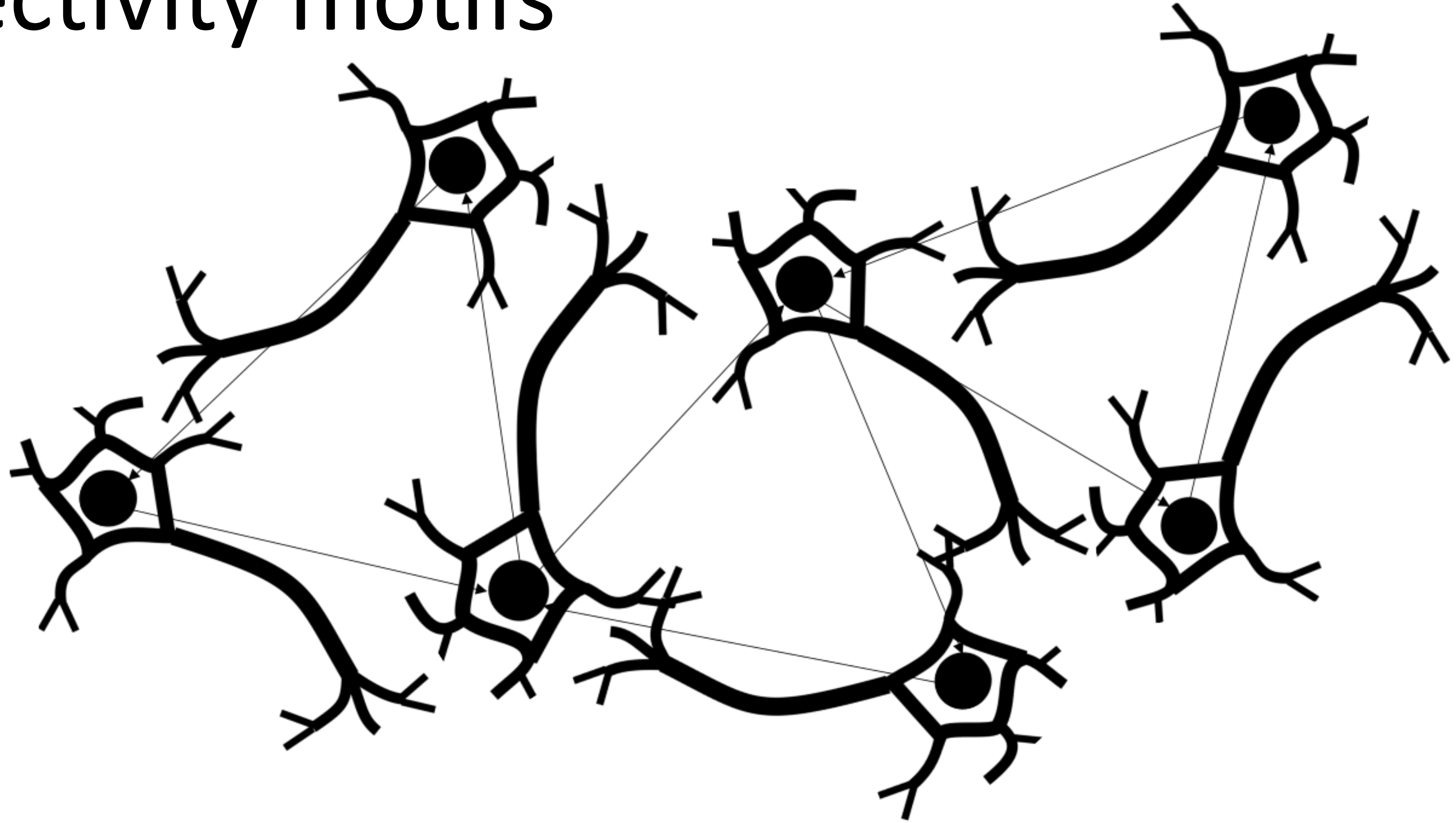


Image Credit: Wikipedia

Slide credit: Jakob Troidl, Hanspeter Pfister

Motif finding

- **Large Networks.** ~57,000 nodes and ~130 million edges.
- **Expensive Computation.** Verifying the existence of a motif in a larger network is NP-complete.
- **Complex 3D structure.** Neurons span long volumes and form complex branching patterns.

- Algorithms:
  - **Ullmann** (2010) which is a recursive backtracking algorithm for solving the subgraph isomorphism problem
  - **Cordella** (2004) another algorithm based on Ullmann's, VF2, which improves the refinement process using different heuristics and uses significantly less memory.

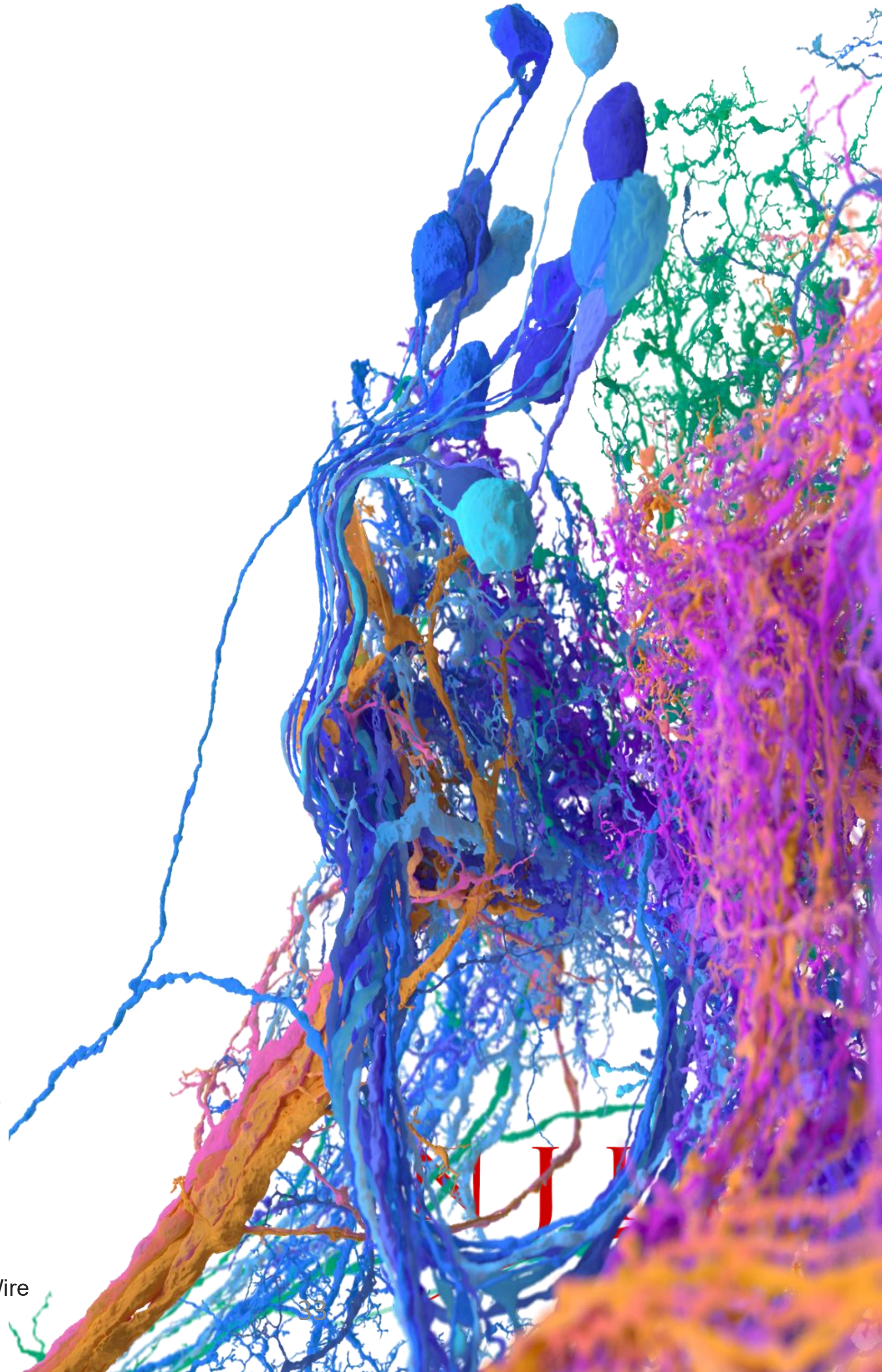
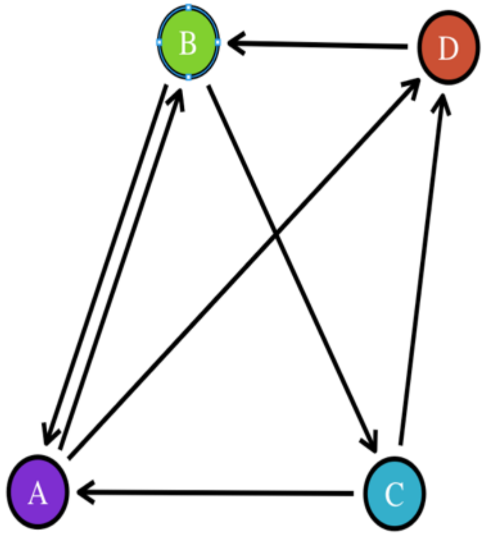


Image Credit: Amy Sterling @ FlyWire

Slide modified from: Jakob Troidl, Hanspeter Pfister

Neuroscientists with to correlate motif connectivity to neuron morpholog



?

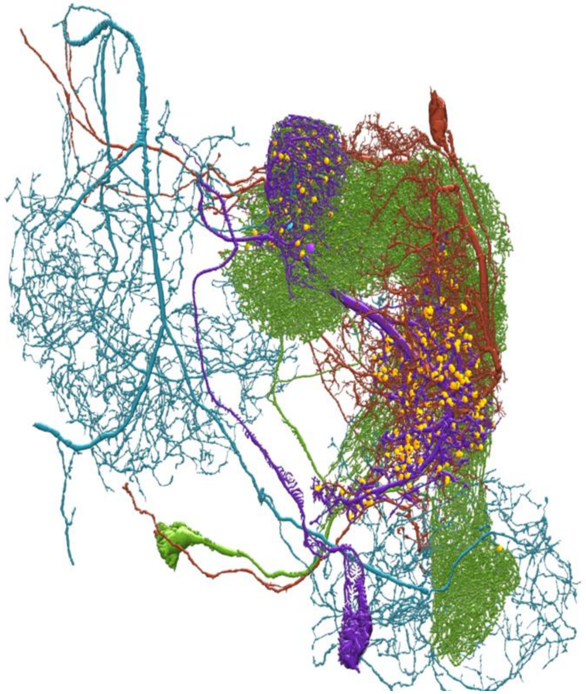
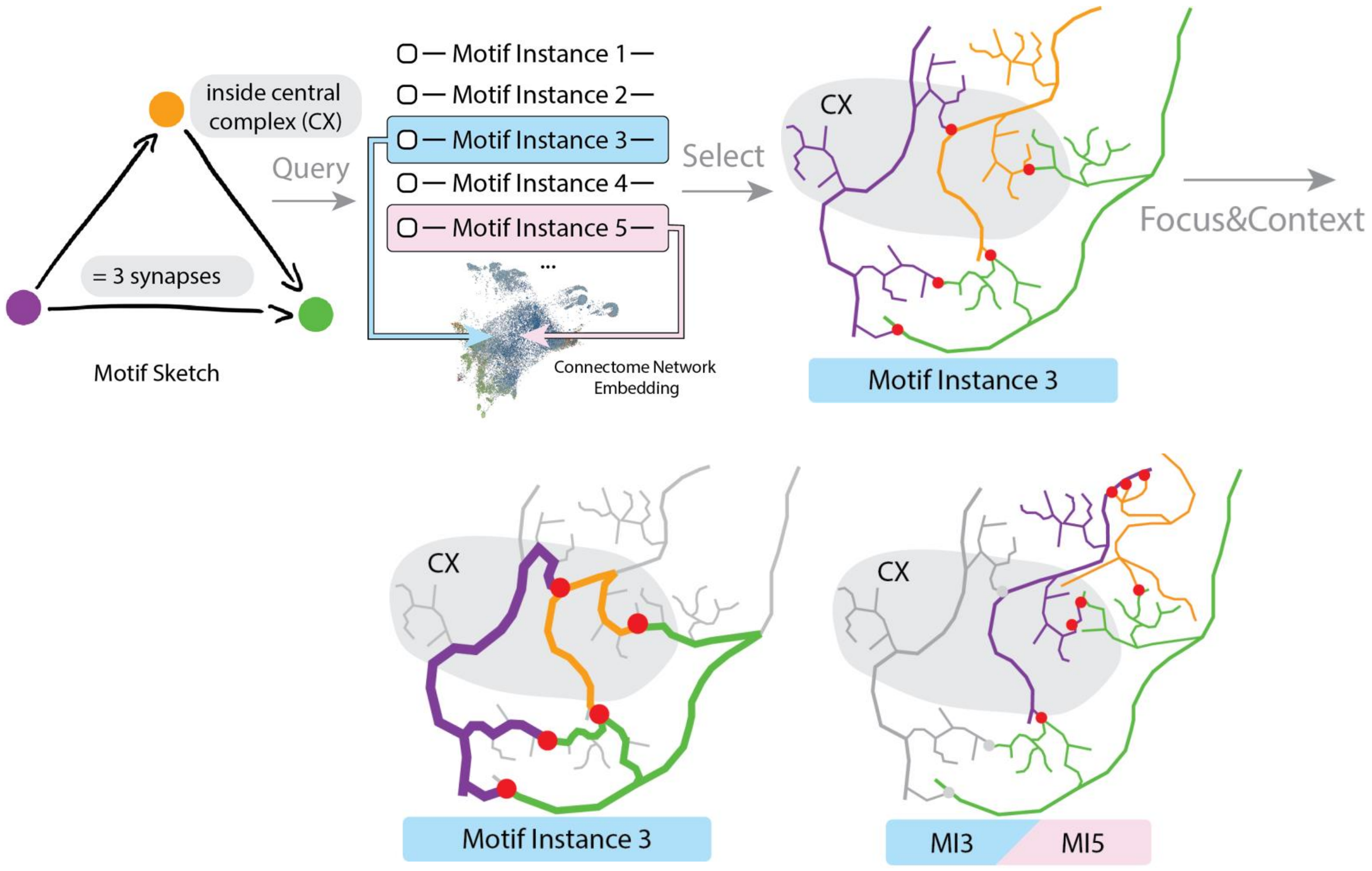


Image credit: Jakob Troidl, Hanspeter Pfister

NJIT New Jersey Institute of Technology

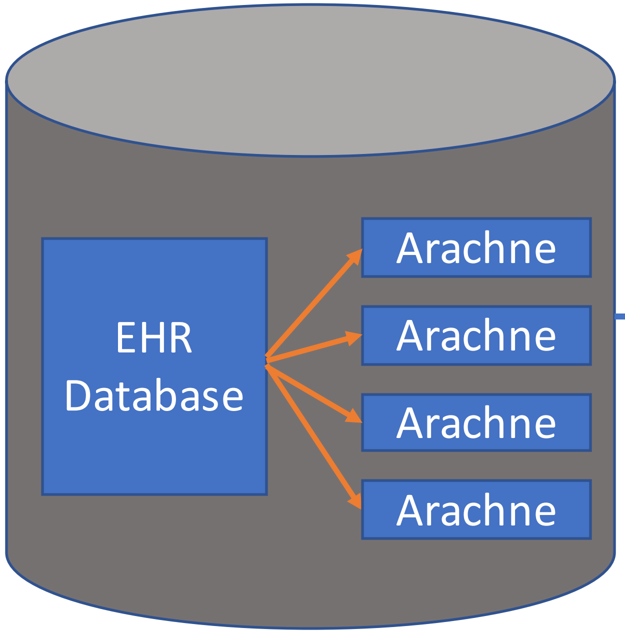


Slide credit: Jakob Troidl, Hanspeter Pfister

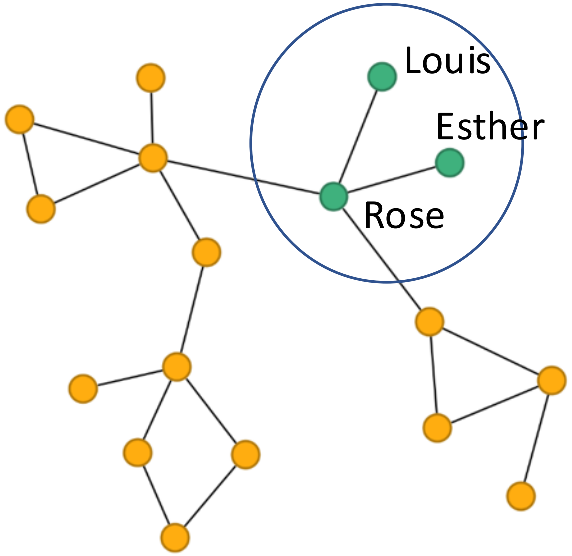
Finding Patterns in Clinical Patient Records

- The adoption of electronic health record (EHR) systems has simultaneously changed clinical practice.
- In data from 2019 and 2021, 96% of general acute care hospitals had adopted EHR[*]

[*] Office of the National Coordinator for Health Information Technology. Adoption of Electronic Health Records by Hospital Service Type 2019-2021, Health IT Quick Stat #60. April 2022.



Data pre-processed by different tasks on multicore Arachne server

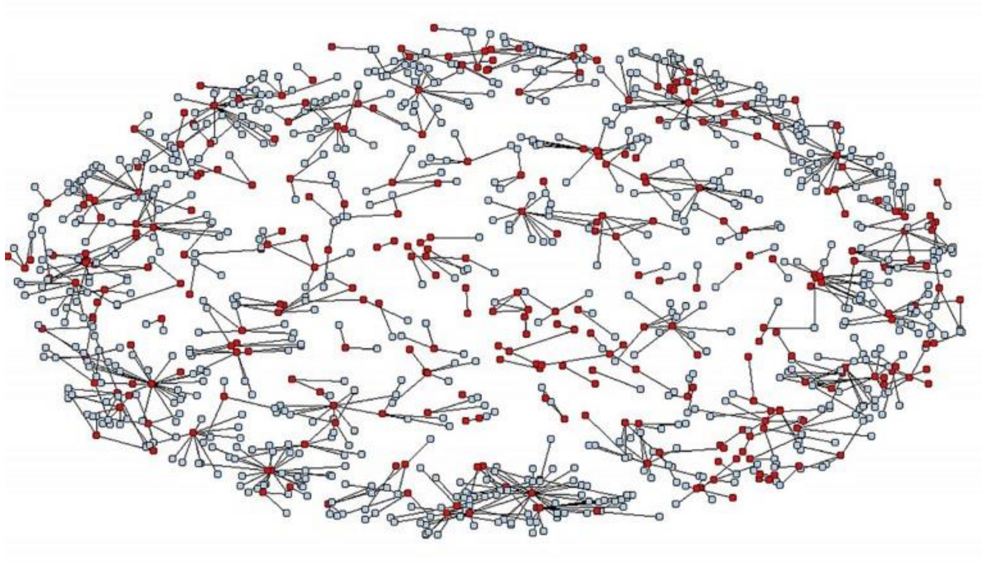


Vertex: Patient
Edge: Shared clinical features

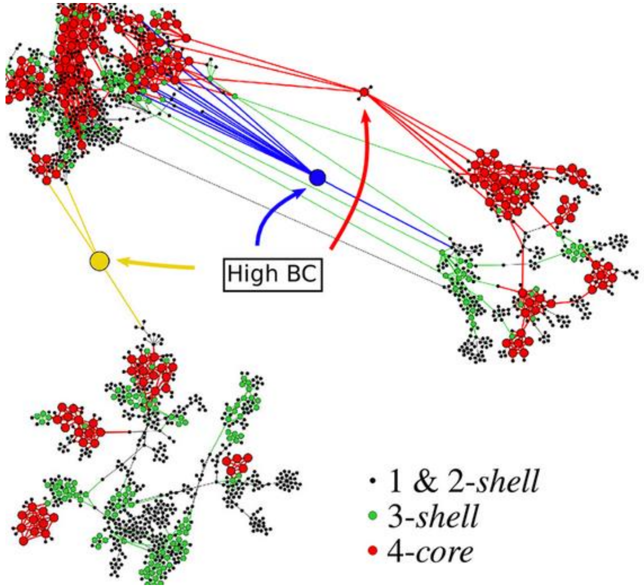
- Utilize **community detection** algorithms to identify groups of vertices.
- These communities may correspond to subpopulations of patients with similar clinical characteristics or disease trajectories.

NJIT New Jersey Institute of Technology

Contact Tracing Networks (COVID, HIV, etc.)



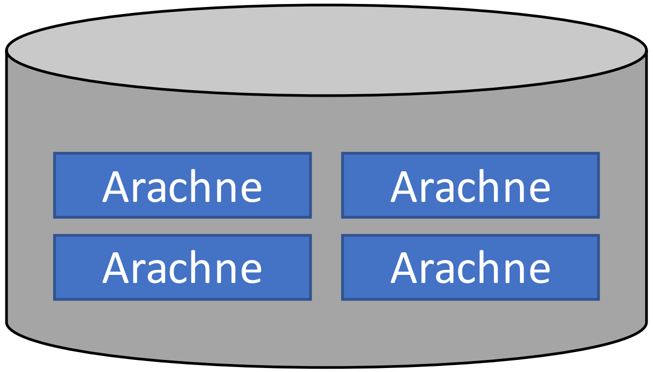
[Cushman, 2020]



[Serafino, Monteiro, et al. 2022]

Usual characteristics of graphs:

- million+ edges and vertices give rise to graphs that take more than 2GB to store in memory
- storage of in-between steps of certain algorithms like Jaccard coefficients can exceed 512GB



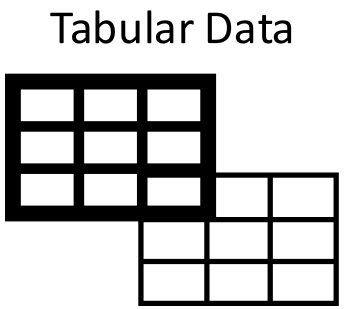
Current:

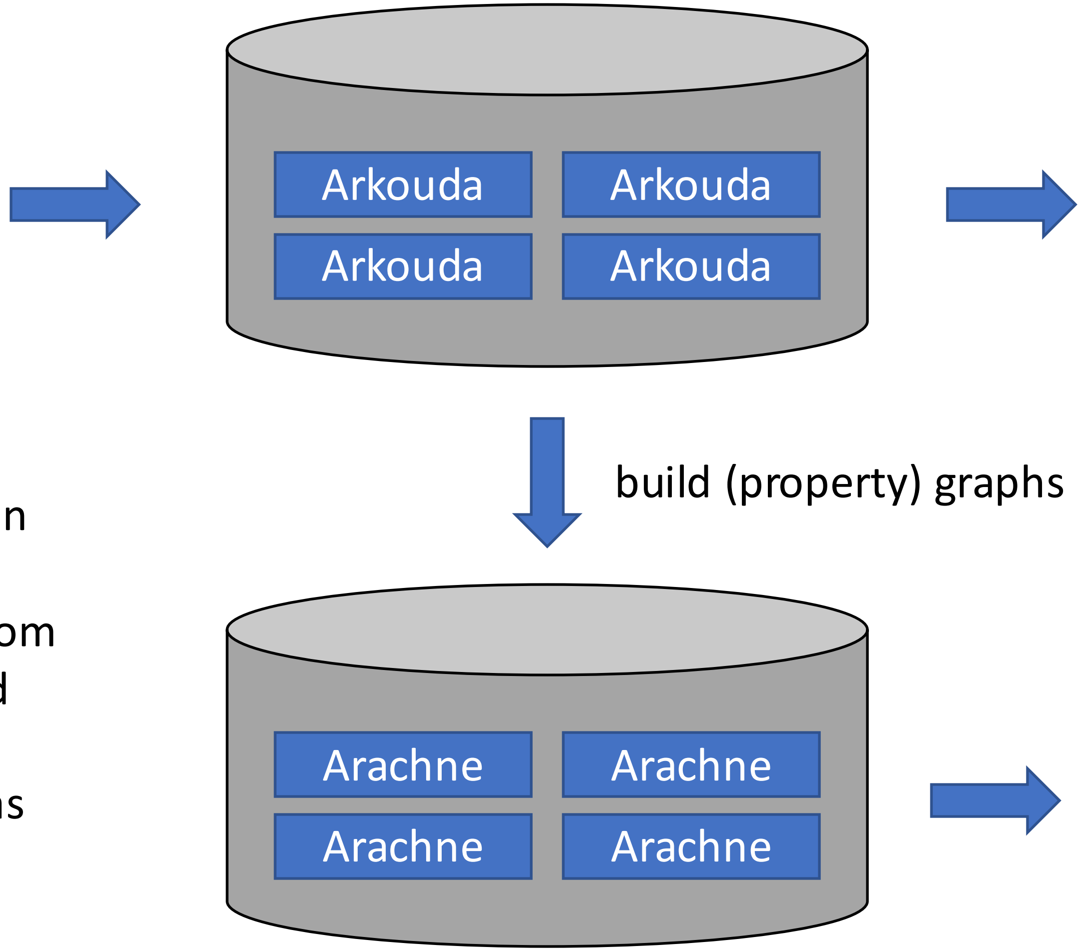
- k-truss
- triangle counting
- triangle centrality

Planned:

- re-implementation of bc
- k-core

Population Health Data Analysis





Analysis on terabytes of tabular data!

- Arachne works with Arkouda as an add-on for graph analysis.
- Data can be taken from Arkouda and created into graphs by specifying columns as edge sources and destinations.

- run graph kernels
- planned: property graph algorithms

New Jersey Institute of Technology

The Arkouda-Arachne Netflow Data Pipeline

IPV4_SRC_ADDR	L4_SRC_PORT	IPV4_DST_ADDR	L4_DST_PORT	PROTOCOL	L7_PROTO	IN_BYTES	OUT_BYTES	IN_PKTS	OUT_PKTS	TCP_FLAGS	FLOW_DURATION_MS	Attack
192.168.100.6	52670	192.168.100.1	53	17	5.21	71	126	1	1	0	4294966	Benign
192.168.100.6	49160	192.168.100.149	444	6	0	217753000	199100	4521	4049	24	4176249	Theft
192.168.100.46	3456	192.168.100.5	80	17	0	8508021	8918372	9086	9086	0	4175916	Benign
192.168.100.3	80	192.168.100.55	8080	6	7	8442138	9013406	9086	9086	0	4175916	Benign

`ak.stick()`

SOURCE ~ DESTINATION

IPV4 source addresses and ports together make up the source vertex of the edge and respectively the same columns for the destination vertex of the edge.

integer id gen

`ak.GroupBy()`
`ak.groupby.broadcast()`

IPV4_SRC	IPV4_SRC_id	IPV4_DST	IPV4_DST_id
192.168.100.6:52670	3473	192.168.100.1:53	3455
192.168.100.6:49160	4234	192.168.100.149:444	3233

To Arachne!

NJIT
New Jersey Institute of Technology

Back-End Storage and Querying

Relationships

SRC	DST	IND	SETS
0	1	0	{1}
0	2	1	{0}
1	0	2	{1}
1	2	3	{0,2}
2	2	4	{0}

Edge Properties

INTS	REALS
REF	REF
REF	REF
REF	REF
REF	REF
REF	REF

- Relationships are stored in sets per edge. User specifies a query, and we search the edge set in a massively parallel manner, probing the sets in amortized constant time.
- Properties are stored split by type and for each edge-type pair that exists, we store an associative domain where we extract the data by simply doing an access edge_prop[col_id].

R-MAP	IND
benign	0
theft	1
DDoS	2

INT

1	2
VAL	VAL

REAL

0
VAL

COL-MAP	IND	TYPE
L7_PROTO	0	FLOAT
IN_BYTES	1	INT
OUT_BYTES	2	INT

- All searching is guaranteed to be $O(m/p)$ since it only involves iterating over the edge set in parallel with each processor.

- All REFS point to arrays that are of the specified column type and that store the key-value pairs for column identifier to data.
- Query hits are returned in a Boolean array specified which edges matched.

Code Example for Python Scripts & Jupyter

```
1. import arkouda as ak
2. import arachne as ar
3.
4. ## Get src and dst from input file.
5. graph = ar.PropGraph()
6. graph.add_edges_from(src,dst)
7.
8. ## Generate relationships_df and edge_properties_df from input file.
9. graph.add_edge_relationships(relationships_df)
10.graph.add_edge_properties(edge_properties_df)
11.
12.## User generates relationships_to_find and property query.
13.returned_edges_rel = graph.query_relationships(relationships_to_find)
14.returned_edges_prop = graph.query_edge_properties(“COLUMN”, 67, “>”)
15.
16.returned_edges = ak.intersect1d(returned_edges_rel, returned_edges_prop)
17.subgraph_src = returned_edges[0]
18.subgraph_dst = returned_edges[1]
19.
20.subgraph = ar.Graph()
21.subgraph.add_edges_from(subgraph_src, subgraph_dst)
22.bfs = ar.bfs_layers(subgraph)
23.cc = ar.connected_components(subgraph)
24.tris = ar.triangles(subgraph)
25.squares = ar.squares(subgraph)
26.## And more!!!!!
```

- Line 6 input is generated from input files from types such as HDF5, CSV, Parquet, etc.
- Lines 9 and 10 input is generated from input files as well.
- Lines 13 and 14 relationships and properties to find are generated by the user.
- Lines 16-19 use Arkouda operations and slicing to get the edges that are returned by both queries.
- Lines 20 and 21 create a new Arachne simple graph with the returned edges of the queries.
- Lines 22-25 run some of the other algorithms available in Arachne!
- Arachne can also return arrays composing of the edges which can be converted to Python lists or NumPy arrays so they can be loaded into NetworkX for further analysis!

New Jersey Institute of Technology

Arachne DI Data Structure [Du et al. 2021]

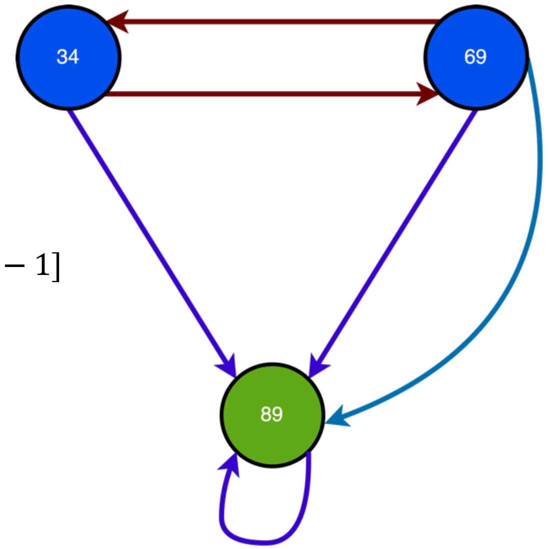
Vertices

	IND	MAP	SEG
L1	0	34	0
	1	69	2
L2	2	89	4
	3		5

Edges

IND	SRC	DST
0	0	1
1	0	2
2	1	0
3	1	2
4	2	2

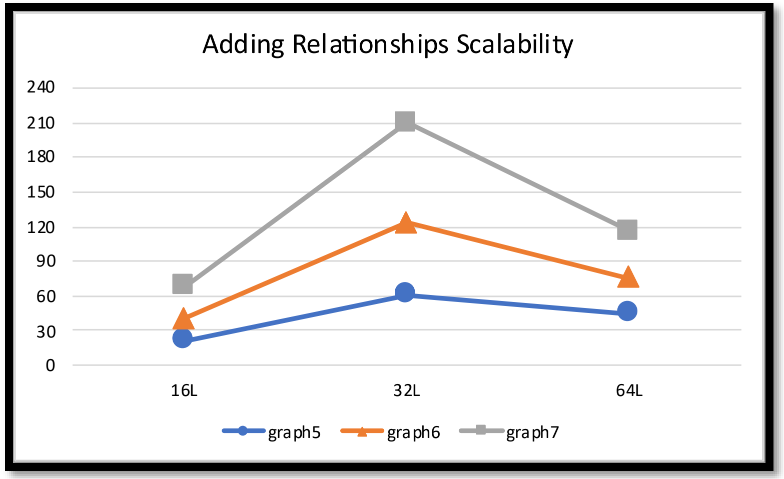
$[SEG[u]..SEG[u+1]-1]$

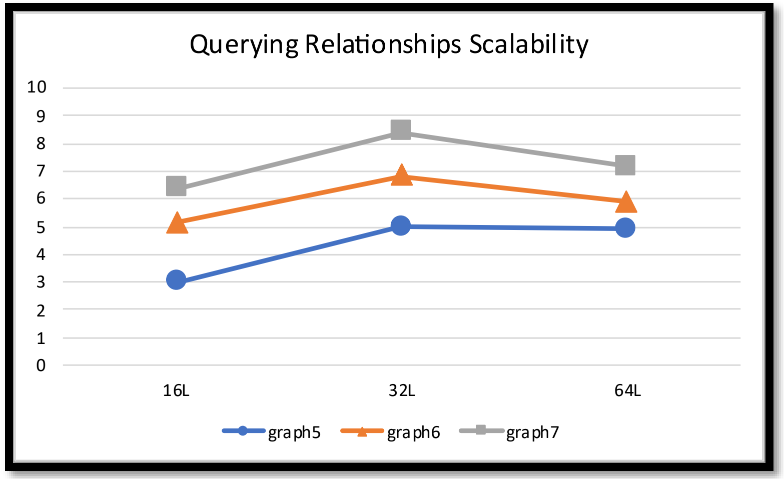


- Allows for simple, compact, **distributable** storage of vertex and edge sets.
- Given an edge index, all vertices that make up that edge are found in **constant time**, avoiding a binary search into SRC (CSR offsets index equivalent).
- MAP allows explicitly storing original vertex labels, returning original graph involves creating arrays and place values of SRC[MAP] and DST[MAP] into new arrays.

Property Graph Results

	n	m
graph5	864,648,454	1,000,000,000
graph6	2,161,664,289	2,500,000,000
graph7	1,408,892,291	5,000,000,000





- Experiments conducted on a cluster where each compute node was composed of 128 cores (64 per AMD EPYC 7713 CPUs), 1TB DDR4 RAM, and an Infiniband HDR 200 GB/s node interconnect.
  - At time of results, some nodes had performance issues, hence the weird elbows.
  - Fifty random relationships were made and randomly assigned to edge indices meaning some edges could be picked more than once and some none at all.
- Querying involved searching for the edges that included three of the fifty relationships, each list performed a set and operation with the search space.

Takeaway: Building a graph of five billion edges takes under 60 seconds, running ETL to insert relationships takes under 4 minutes, and querying it under 10 seconds.

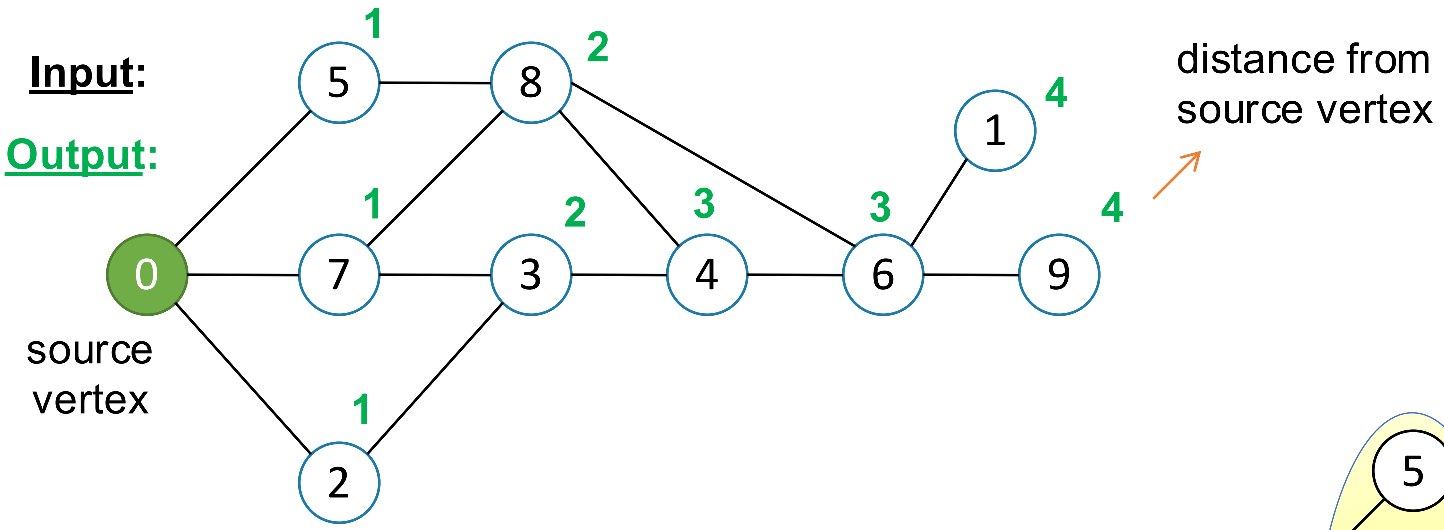
New Jersey Institute of Technology

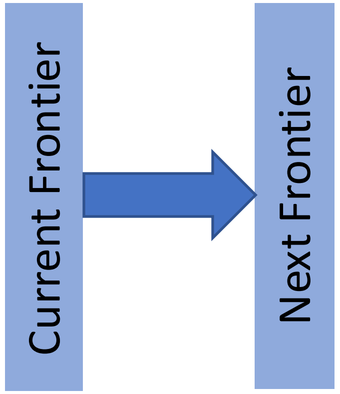
Graph Algorithms in Arachne

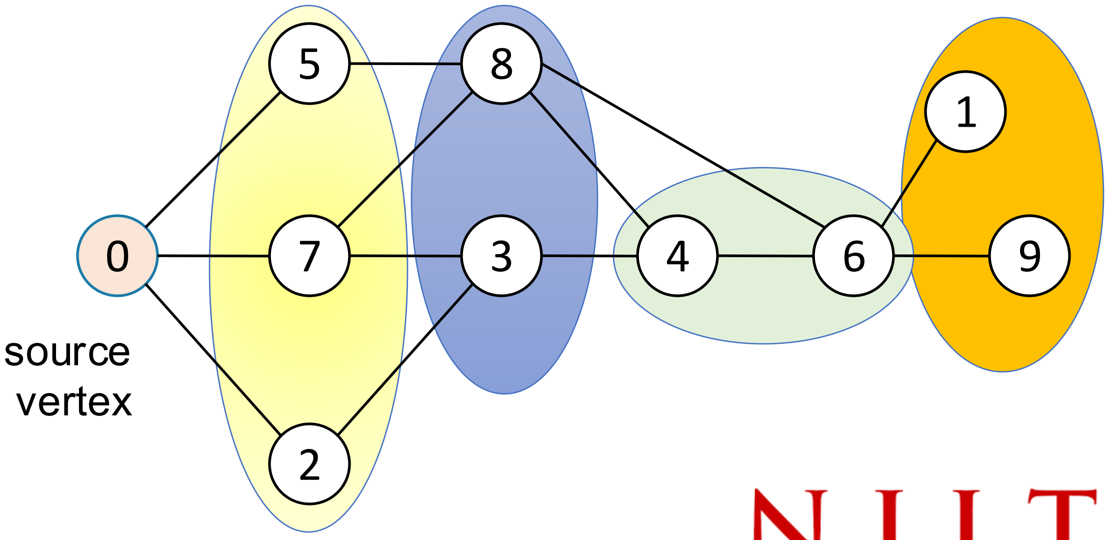
- **Breadth-first search (BFS)** [Du, Alvarado Rodriguez, Bader 2021]
  Returns an array of size n with how many hops away some vertex v is from an initial vertex u.
- **Connected components**
  Returns an array of size n where all vertices who belong to the same component have the same value x. The value of x is the label of the largest vertex in the component.
- **Triangle counting** [Du, Alvarado Rodriguez, Patchett, Bader 2021]
  Returns the number of triangles in a graph.
- **Truss Analytics** [Du, Patchett, Bader 2021][Du, Patchett, Alvarado Rodriguez, Li, Bader 2022]
  K-truss returns every edge in the truss where each edge must be a part of $k-2$ triangles that are made up of nodes in that truss. Max truss returns the maximum k. Truss decomposition returns the maximum k for each edge.
- **Square counting** [Burkhardt, Harris 2023]
  Returns the number of four-cycles in the graph.
- **Triangle centrality** [Patchett, Du, Bader 2022][Patchett, 2022]
  Returns an array of size n with the proportion of triangles centered at a vertex v.
- **Subgraph isomorphism** [Dindoost, Bader, 2023, in progress]
  Finds instances of a pattern in a larger graph.

New Jersey Institute of Technology

Shared-Memory Parallel Breadth-First Search





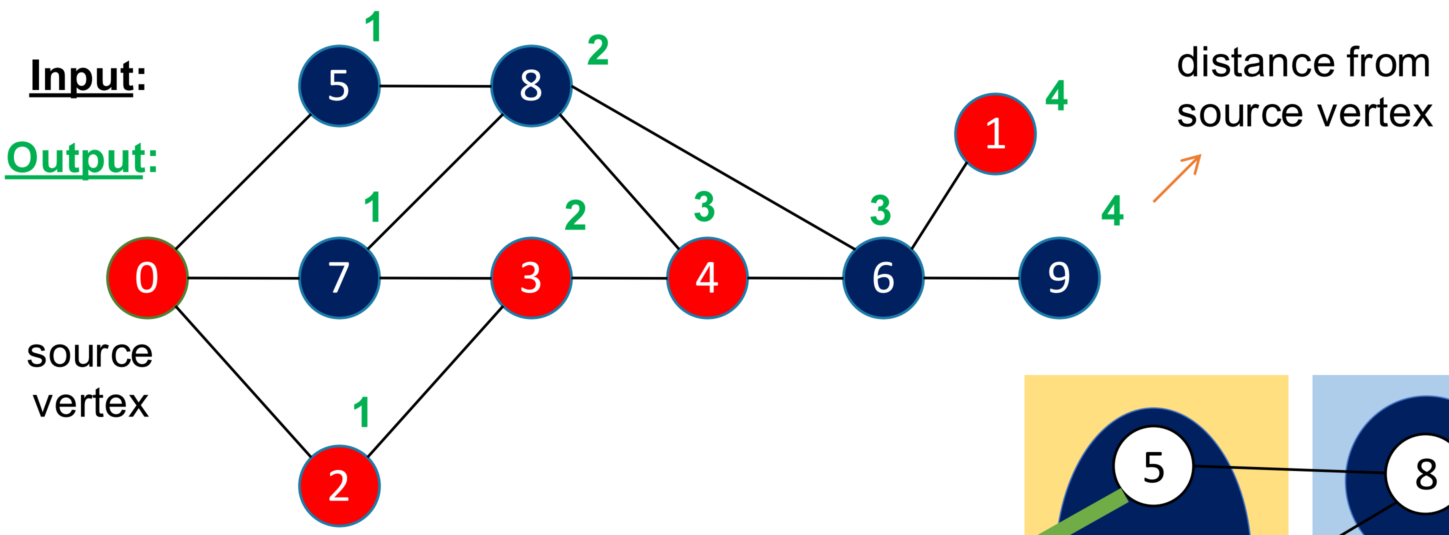


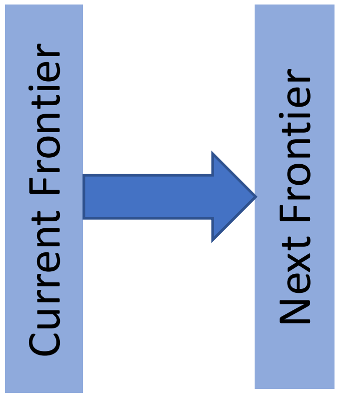
Output: D = [0, 4, 1, 2, 3, 1, 3, 1, 2, 4]

New Jersey Institute of Technology

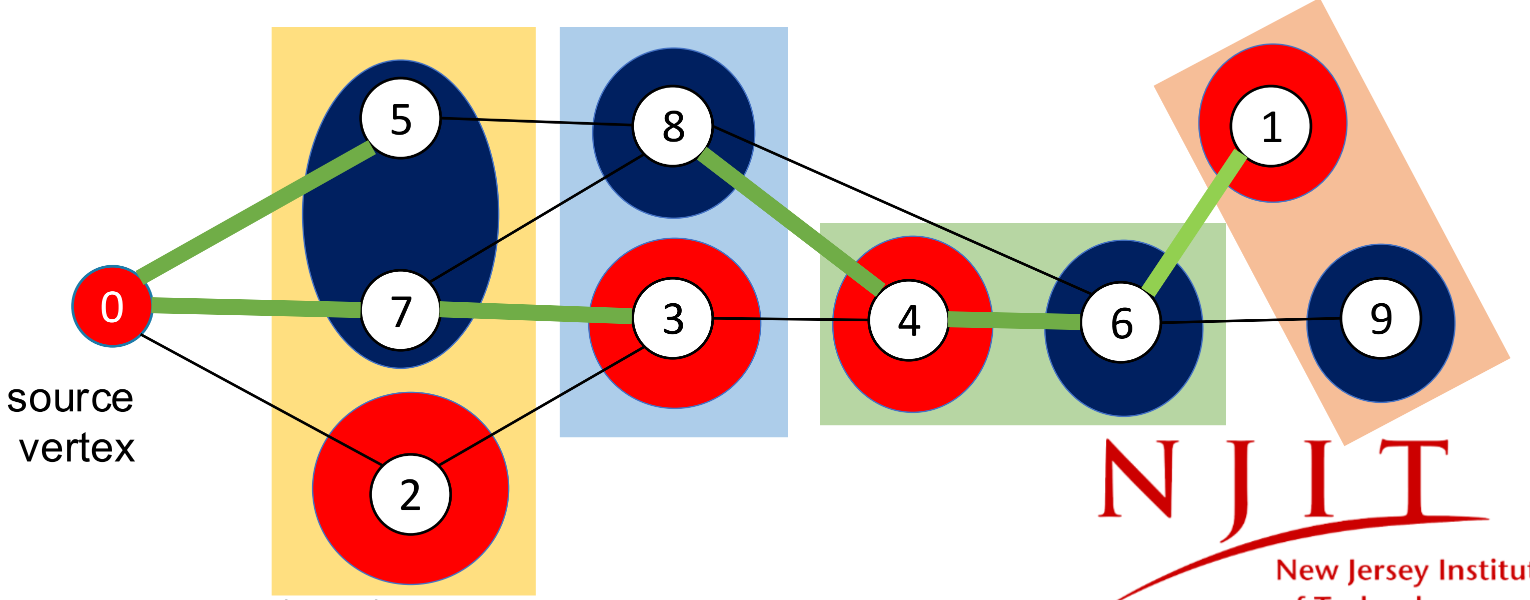
Distributed-Memory Parallel Breadth-First Search

Assume our edge list is split down the middle, then the neighborhood of some vertices will live on one compute node while the rest live on another compute node.





Output: D = [0, 4, 1, 2, 3, 1, 3, 1, 2, 4]



Any cross-color expansions are writes across the network; fine-grained writes hold up execution, large coarse-grained writes are better.

New Jersey Institute of Technology

Breadth-First Search Communication Volume Results

delaunayn20	get			put		
locale	di	di-norev	di-agg-ls	di	di-norev	di-agg-ls
0	15672640	7873842	639827	5629422	2749193	138070
1	15834332	7939017	687156	1952226	1016946	127936
2	15715554	7722659	226754	1942839	962031	45217
3	15817879	7723971	226880	1951313	962201	45060
4	15964559	7724880	226691	1961552	962199	51217
5	15739226	7726504	230024	1940688	962439	52714
6	15569450	7727678	229096	1925536	962680	51977
7	15341933	7736094	225083	1904757	963418	48413

di: 84 seconds

di-agg-ls: 3.36 seconmds

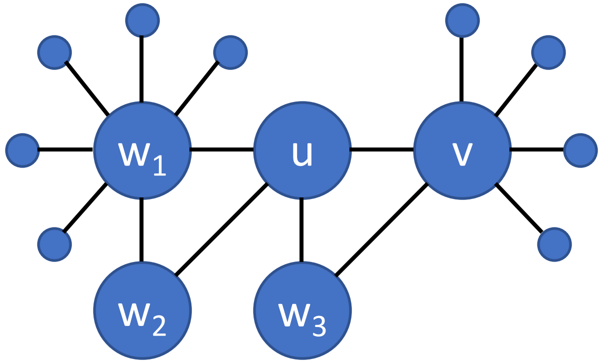
delaunayn20 is a graph with 3 million edges and a large diameter

Takeaway: Aggregating writes drastically reduces communication volumes, improving performance, all done easily through Chapel by adapting aggregators for different uses.

NJIT New Jersey Institute of Technology

Minimum Search Triangle Counting

1. Given an edge (u, v) we assume that $|Adj(u)| \leq |Adj(v)|$.
2. Then, for $\forall w \in Adj(u)$ we spawn $|Adj(u)|$ - 1 parallel threads to check if we can form a complete triangle with (u, v, w).
3. If $|Adj(w)| < |Adj(v)|$ we will check if $v \in Adj(w)$, else, we check if $w \in Adj(v)$.



Adj(x)	Value
Adj(u)	4
Adj(v)	6
Adj(w1)	7
Adj(w2)	2
Adj(w3)	2

Thread w_1: search for w_1 in $Adj(v)$, no match, kill.

Thread w_2: search for v in $Adj(w_2)$, no match, kill.

Thread w_3: search for v in $Adj(w_3)$, match! Increment count.

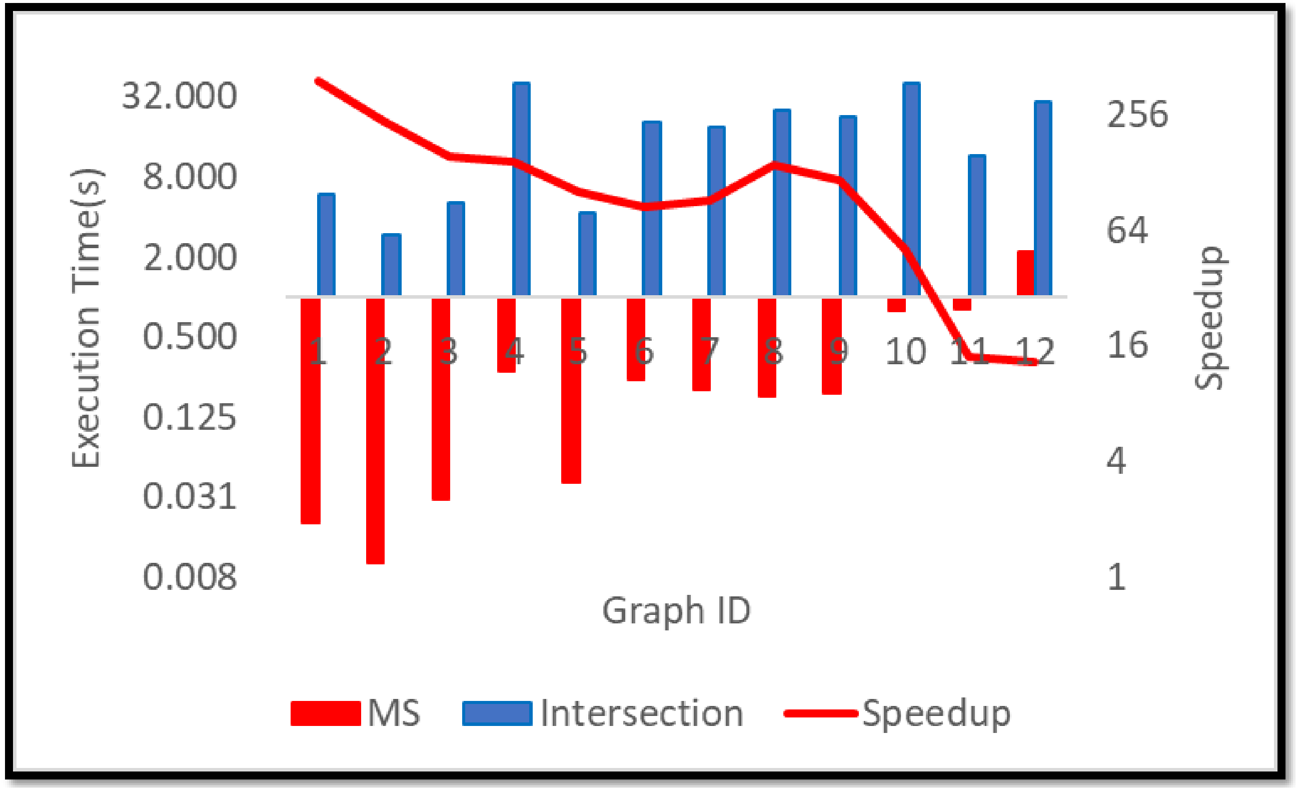
NJIT New Jersey Institute of Technology

Minimum Search Triangle Counting Operation Count Comparison

- Assume $|Adj_u| < |Adj_v|$ and we spawn threads for every $w \in |Adj_u|$
  - Minimum search: $\max_{w \in Adj_u} \log_2(\min(|Adj_w|, |Adj_v|))$
  - List Intersection: $\log_2(|Adj_v|)$
- Say we have the following information for our vertices:
  - $|Adj_u| = 4$ and $|Adj_v| = 1024$
  - For every w in Adj_u, $|Adj_w| \leq 8$
- **List intersection:** 4 threads amounting to $\lceil \log_2 1024 \rceil = 10$ operations each.
- **Minimum search:** 4 threads amounting to $\lceil \log_2 8 \rceil = 3$ operations each.

New Jersey Institute of Technology

Triangle Counting Results



- Our method outperforms with the Conte method with a highest speedup of 385.8 and an average speedup of 128.

Takeaway: Truss decomposition with minimum search triangle counting outperforms a C++ method coded with OpenMP, with SSE-Acceleration, binary searching on adjacency list, and no atomic operations.

Graphs used were a variety of real-world graphs available for view in paper.

NJIT New Jersey Institute of Technology

Major Contributions

- Arachne, a large-scale graph analysis framework, extends Arkouda for productive graph analysis. Arachne is built on a novel sparse graph data structure.
- Arachne leverages productivity through Python with high performance backed by Chapel.
- Arachne, Arkouda, Chapel are all open-source.
  - https://github.com/Bears-R-Us/arkouda-njit
  - https://github.com/Bears-R-Us/arkouda
  - https://github.com/chapel-lang/chapel
- Experimental results on real-world and synthetic graphs demonstrate that Arachne works for graphs with billions of edges.

NJIT
New Jersey Institute of Technology

Publications

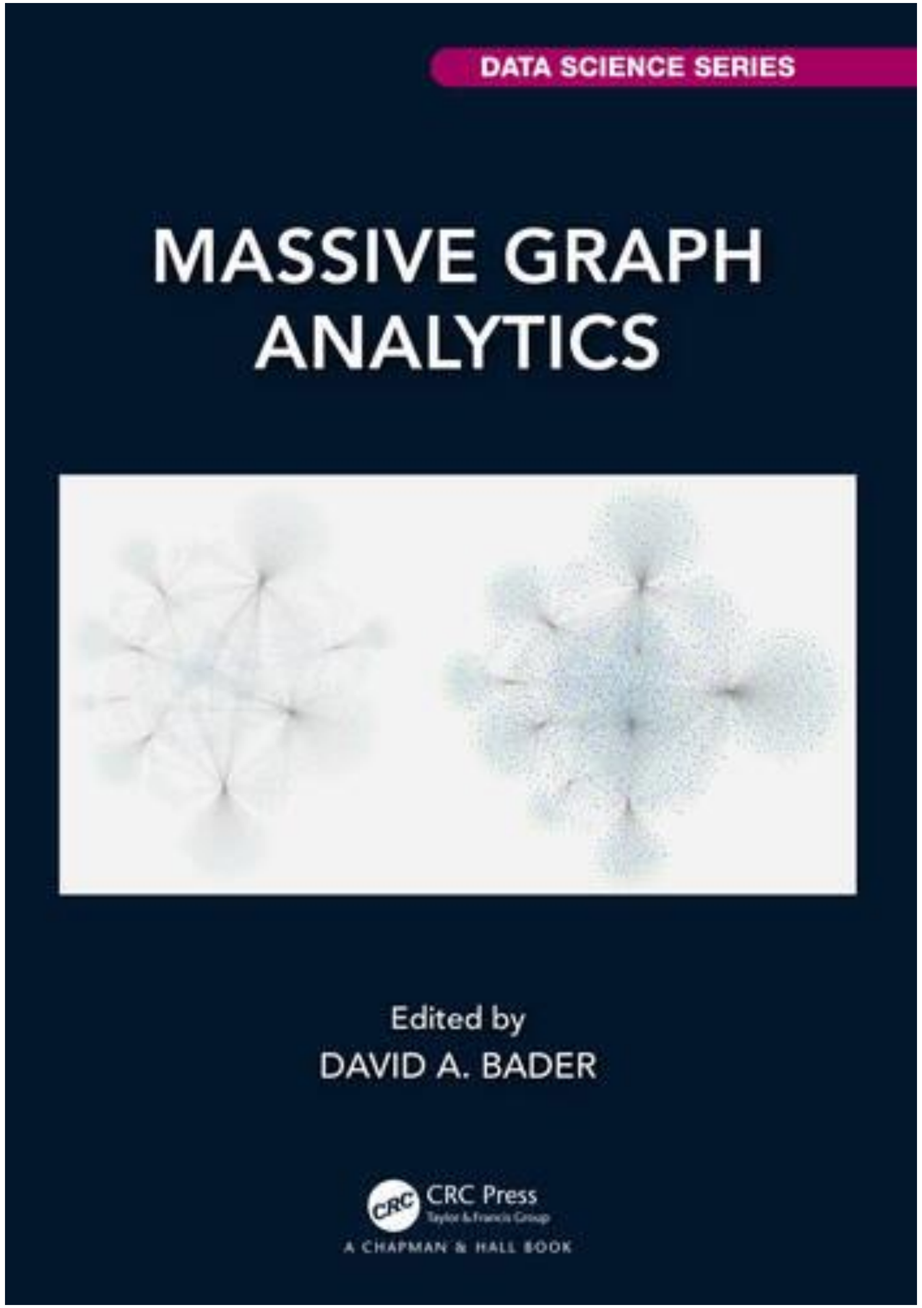
- **Oliver Alvarado Rodriguez, Zhihui Du, Joseph Patchett, Fuhuan Li, David Bader (2022). Arachne: An Arkouda Package for Large-Scale Graph Analytics. IEEE HPEC.**
- Oliver Alvarado Rodriguez, Fernando Vera Buschmann, Zhihui Du, David Bader (2023). Property Graphs in Arachne. IEEE HPEC.
- Soroush Vahidi, Baruch Schieber, Zhihui Du, David Bader (2023). Parallel Longest Common SubSequence Analysis In Chapel. IEEE HPEC.
- Joseph Patchett, Zhihui Du, Fuhuan Li, David Bader (2022). Triangle Centrality in Arkouda. IEEE HPEC.
- Zhihui Du, Oliver Alvarado Rodriguez, David Bader (2021). Large Scale String Analytics In Arkouda. IEEE HPEC.
- Zhihui Du, Oliver Alvarado Rodriguez, David Bader (2021). Enabling Exploratory Large Scale Graph Analytics through Arkouda. IEEE HPEC.
- Joseph Patchett, Zhihui Du, David Bader (2021). K-Truss Implementation in Arkouda (Extended Abstract). IEEE HPEC.
- Zhihui Du, Oliver Alvarado Rodriguez, Joseph Patchett, David Bader (2021). Interactive Graph Stream Analytics in Arkouda. Algorithms.
- Zhihui Du, Oliver Alvarado Rodriguez, David A. Bader, Michael Merrill, William Reus (2021). Exploratory Large Scale Graph Analytics in Arkouda. CHIUW.

New Jersey Institute of Technology

Conclusions & Further Work

- We can design and develop high performance graph analysis algorithms using Arkouda/Chapel quickly and efficiently.
- We plan to work on optimizing all current methods to work as efficiently as possible in single locale and multi locale environments.
- We plan to implement new novel algorithms such as stringology, a communication-efficient triangle counting, large-scale community detection, and machine learning.

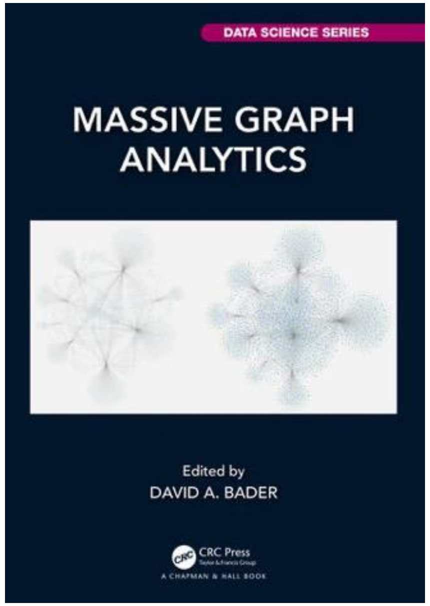
NJIT
New Jersey Institute of Technology



Massive Graph Analytics
Edited By David A. Bader
Copyright Year 2022
ISBN 9780367464127
Published July 20, 2022 by Chapman & Hall
616 Pages 207 B/W Illustrations

NJIT
New Jersey Institute of Technology

Chapters



<u>**Algorithms: Search and Paths**</u>
A Work-Efficient Parallel Breadth-First Search Algorithm (or How to Cope With the Nondeterminism of Reducers
Charles E. Leiserson and Tao B. Schardl
Multi-Objective Shortest Paths
Stephan Erb, Moritz Kobitzsch, Lawrence Mandow , and Peter Sanders

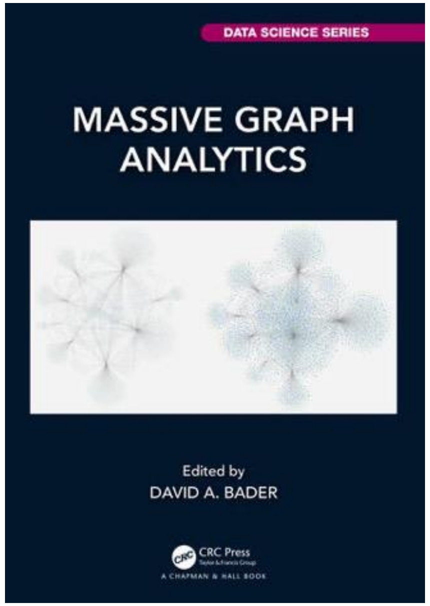
<u>**Algorithms: Structure**</u>
Multicore Algorithms for Graph Connectivity Problems
George M. Slota, Sivasankaran Rajamanickam, and Kamesh Madduri
Distributed Memory Parallel Algorithms for Massive Graphs
Maksudul Alam, Shaikh Arifuzzaman, Hasanuzzaman Bhuiyan, Maleq Khan, V.S. Anil Kumar, and Madhav Marathe
Efficient Multi-core Algorithms for Computing Spanning Forests and Connected Components
Fredrik Manne, Md. Mostofa Ali Patwary
Massive-Scale Distributed Triangle Computation and Applications
Geoffrey Sanders, Roger Pearce, Benjamin W. Priest, Trevor Steil

<u>**Algorithms and Applications**</u>
Computing Top-k Closeness Centrality in Fully-dynamic Graphs
Eugenio Angriman, Patrick Bisenius, Elisabetta Bergamini, Henning Meyerhenke
Ordering Heuristics for Parallel Graph Coloring
William Hasenplaugh, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson
Partitioning Trillion Edge Graphs
George M. Slota, Karen Devine, Sivasankaran Rajamanickam, Kamesh Madduri
New Phenomena in Large-Scale Internet Traffic
Jeremy Kepner, Kenjiro Cho, KC Claffy, Vijay Gadepally, Sarah McGuire, Peter Michaleas, Lauren Milechin
Parallel Algorithms for Butterfly Computations
Jessica Shi and Julian Shun

<u>**Models**</u>
Recent Advances in Scalable Network Generation
Manuel Penschuck, Ulrik Brandes, Michael Hamann, Sebastian Lamm, Ulrich Meyer, Ilya Safro, Peter Sanders, and Christian Schulz
Computational Models for Cascades in Massive Graphs: How to Spread a Rumor in Parallel
Ajitesh Srivastava, Charalampos Chelmis, Viktor K. Prasanna
Executing Dynamic Data-Graph Computations Deterministically Using Chromatic Scheduling
Tim Kaler, William Hasenplaugh, Tao B. Schardl, and Charles E. Leiserson

<u>**Frameworks and Software**</u>
Graph Data Science Using Neo4j
Amy E. Hodler, Mark Needham
The Parallel Boost Graph Library 2.0
Nicholas Edmonds and Andrew Lumsdaine
RAPIDS cuGraph
Alex Fender, Bradley Rees, Joe Eaton
A Cloud-based approach to Big Graphs
Paul Burkhardt and Christopher A. Waring
Introduction to GraphBLAS
Jeremy Kepner, Peter Aaltonen, David Bader, Aydin Buluc, Franz Franchetti, John Gilbert, Dylan Hutchinson, Manoj Kumar, Andrew Lumsdaine, Henning Meyerhenke, Scott McMillian, Jose Moreira, John D. Owens, Carl Yang, Marcin Zalewski, and Timothy G. Mattson
Graphulo: Linear Algebra Graph Kernels
Vijay Gadepally, Jake Bolewski, Daniel Hook, Shana Hutchison, Benjamin A Miller, Jeremy Kepner
Interactive Graph Analytics at Scale in Arkouda
Zhihui Du, Oliver Alvarado Rodriguez, Joseph Patchett, and David A. Bader

85 Contributors



Peter Aaltonen
Maksudul Alam
Md. Mostofa Ali Patwary
Eugenio Angriman
V.S. Anil Kumar
William Arcand
Shaikh Arifuzzaman
Elisabetta Bergamini
William Bergeron
David Bestor
Hasanuzzaman Bhuiyan
Patrick Bisenius
Ulrik Brandes
Aydin Buluc
Paul Burkhardt
Chansup Byun
Charalampos Chelmis
Kenjiro Cho
K.C. Claffy
Karen Devine
Zhihui Du
Joe Eaton
Nicholas Edmonds
Stephan Erb
Alex Fender
Franz Franchetti
Vijay Gadepally
John Gilbert
Michael Hamann
William Hasenplaugh
Amy E. Hodle
Michael Houle
Matthew Hubbell
Shana Hutchison
Hayden Jananthan
Michael Jones
Tim Kaler
Jeremy Kepner
Maleq Khan
Moritz Kobitzsch
Manoj Kumar
Sebastian Lamm
Charles E. Leiserson
Andrew Lumsdaine
Kamesh Madduri
Lawrence Mandow
Fredrik Manne
Madhav V. Marathe
Timothy G. Mattson
Sarah McGuire
Scott McMillan
Ulrich Meyer
Henning Meyerhenke
Peter Michaleas
Lauren Milechi
Jose Moreira
Mark Needham
John D. Owens
Joseph Patchett
Roger Pearce
Manuel Penschuck
Viktor K. Prasanna
Benjamin W. Priest
Andrew Prou
Sivasankaran Rajamanickam
Bradley Rees
Albert Reuther
Oliver Alvarado Rodriguez
Antonio Rosa
Ilya Safro
Siddharth Samsi
Geoffrey Sanders
Peter Sanders
Tao B. Schardl
Christian Schulz
Jessica Shi
Julian Shun
George M. Slota
Ajitesh Srivastava
Trevor Steil
Christopher A. Waring
Carl Yang
Charles Yee
Marcin Zalewski

Bader Research Lab

Research Faculty:

Zhihui Du

PhD Students:

Fatemeh Ramezani Khozestani

Fernando Vera Buschmann (Fulbright Scholar)

Fuhuan Li

Mohammad Dindoost

Oliver Alvarado Rodriguez

Asha Saxena

Soroush Vahidi (co-advised with Baruch Schieber)

Undergraduates:

Vanshika Agrawal

Palina Paulichenka

Davor Petrovikj

M.S. students:

Antonita Rachael

Naren Khatwani

Tanima Majumder

Current & Past High School Interns:

Pranhav Sundararajan

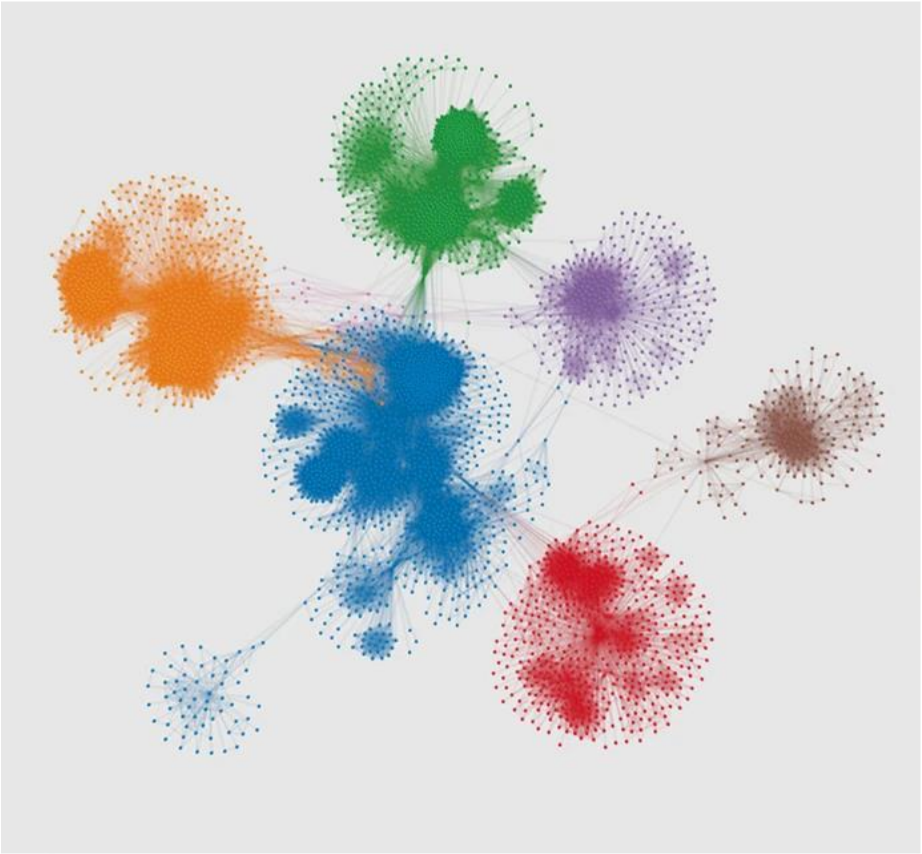
Anya Ganeshan

Sounak Bagchi

NJIT New Jersey Institute of Technology

Community Detection

- In complex networks, nodes cluster and form relatively dense groups – often called communities.
- Community detection is a fundamental graph algorithm with practical applications like ***fraud detection*** in Fintech and ***identity and access management*** in social networks



Vertices are Facebook users and edges represent Facebook friendships. Communities, represented by different colors.

Image credit: Fortunato, S., Newman, M.E.J. 20 years of network community detection. *Nat. Phys.* **18**, 848–850 (2022).

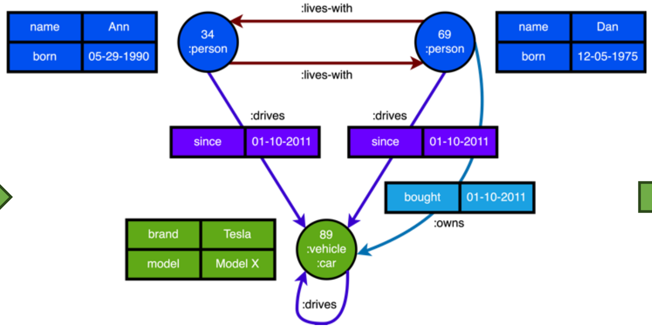
NJIT New Jersey Institute of Technology

Open-Source Massive-Scale (Property) Graph Analytics in Python with Arachne+Arkouda powered by Chapel

id	label	name	born	brand	model
34	person	Ann	1990	NULL	NULL
69					
89					
89					

src id	dst id	relationship	since	bought
34	69	lives-with	NULL	NULL
69	34	lives-with	NULL	NULL
34	89	drives	2011	NULL
69	89	drives	2011	NULL
69	89	owns	NULL	2011
89	89	drives	NULL	NULL

Load in terabytes-sized CSVs, HDF5s, Parquets, etc.



Convert tabular data to graph format with all data closely maintained with vertex and edges.

- query_node_labels()
- query_edge_relationships()
- query_node_properties()
- query_edge_properties()

Extract (multiple) subgraphs by querying on attributes.

- bfs_layers()
- subgraph_isomorphism()
- square_counting()
- and more.....!

Perform further analysis! Convert to NetworkX (if small enough).

```python
import arkouda as ak
import arachne as ar

## Get src and dst from input file.

graph = ar.PropGraph()
graph.add_edges_from(src,dst)

## Generate label_df and relationships_df from input file.

graph.add_node_labels(label_df)
graph.add_edge_relationships(relationships_df)

## User generates labels_to_find and relationships_to_find.
returned_nodes = graph.query_labels(labels_to_find)
returned_edges = graph.query_relationships(relationships_to_find)

subgraph_src = ak.in1d(returned_edges[0], returned_nodes)
subgraph_dst = ak.in1d(returned_edges[1], returned_nodes)

kept_edges = subgraph_src & subgraph_dst

subgraph_src = subgraph_src[kept_edges]
subgraph_dst = subgraph_dst[kept_edges]

subgraph = ar.Graph()
subgraph.add_edges_from(subgraph_src, subgraph_dst)
## Run some new operations on subgraph! Reference our HPEC22 paper ☺
```

Easily usable through NetworkX-like API. Data exchangeable between NetworkX, NumPy, SciPy, etc.

Meta	Distributed Arrays
Meta	Distributed Graphs
Meta	Distributed Strings
MPP, SMP, Cluster, Laptop, etc.	

Runs on any hardware, data stays in the back-end, user calls API through Pythpm: powerful and productive. Server can run on supercomputers, Python API usable locally.

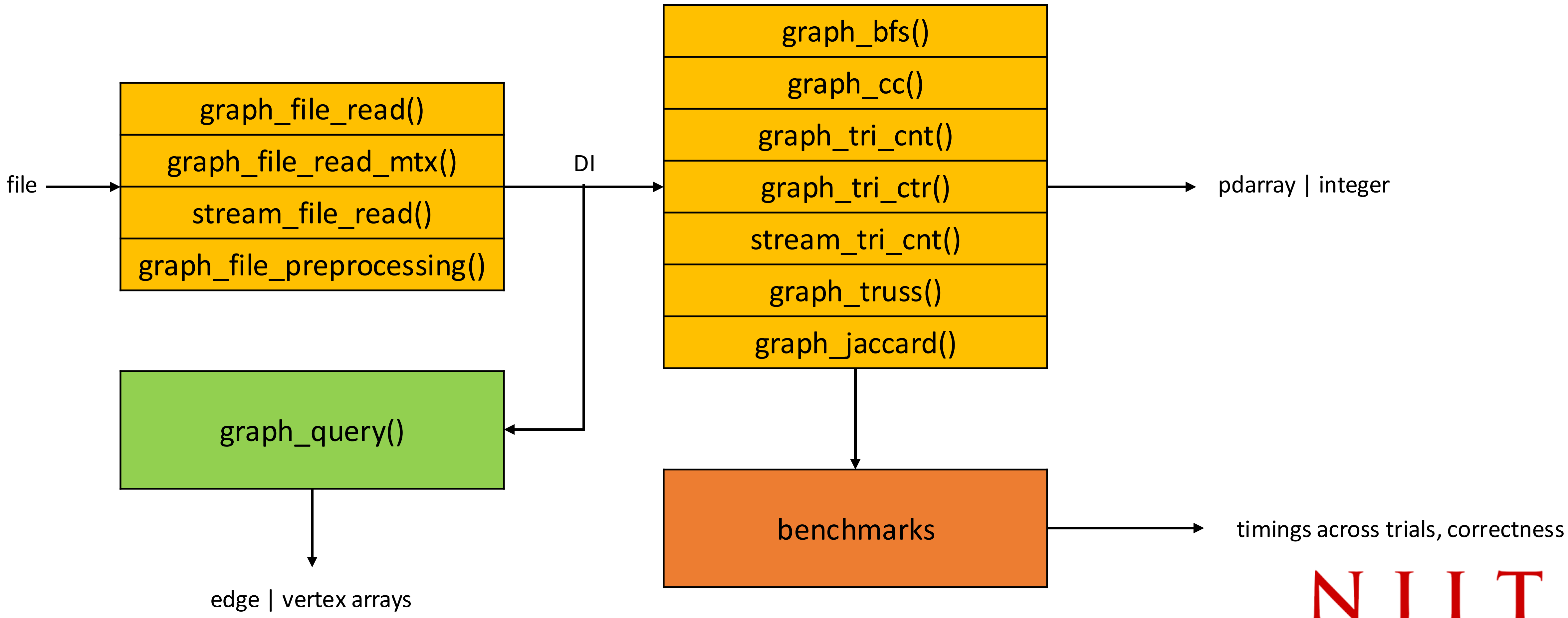
Kernel	Time(s)
Loading graph into memory	5.4
Node label ETL and storage	1.87
Edge label ETL and storage	80.29
Node querying	1.01
Edge querying	1.39
Subgraph building	1.22
Breadth-first search	2.45

Performance analyzing a 10-billion edge graph on 16 compute nodes with 128 cores each and 1TB memory. Subgraph generated contained 1-million edges. BFS executed from the 5 highest degree vertices and the average was taken.

OPEN SOURCE: https://github.com/Bears-R-Us/arkouda-njit
PUBLICATIONS: https://davidbader.net/publication/ filter with “Arkouda” or “Arachne”

NJIT New Jersey Institute of Technology

Modules of Arachne



New Jersey Institute of Technology

Graphs for Testing

Experiments were conducted on a high-performance server with 2 x Intel Xeon E5-2650 v3 @ 2.30GHz CPUs with 10 cores per CPU and a RAM capacity of 512GB.

values found by our algorithms (CCs, Triangles, Max K)

Real-world

Graphs	Edges	Vertices	CCs	Triangles	Max K
ca-GrQc	14484	5242	354	48260	44
ca-HepTh	25973	9877	427	28339	32
as-caida20071105	53381	26475	1	36365	16
facebook_combined	88234	4039	1	1612010	97
ca-CondMat	93439	23133	567	173361	26
ca-HepPh	118489	12008	276	3358499	239
email-Enron	183831	36692	1065	727044	22
ca-AstroPh	198050	18772	289	1351441	57
loc-brightkite_edges	214078	58228	547	494728	43
soc-Epinions1	405740	75879	2	1624481	33
amazon0601	2443408	403394	7	3986507	11
com-Youtube	2987624	1134890	1	3056386	19
friendster	1806067135	65608366	1	4173724142	129

few vertices, outperforms some algorithms less edges but more vertices.

delaunayn10 - delaunayn19

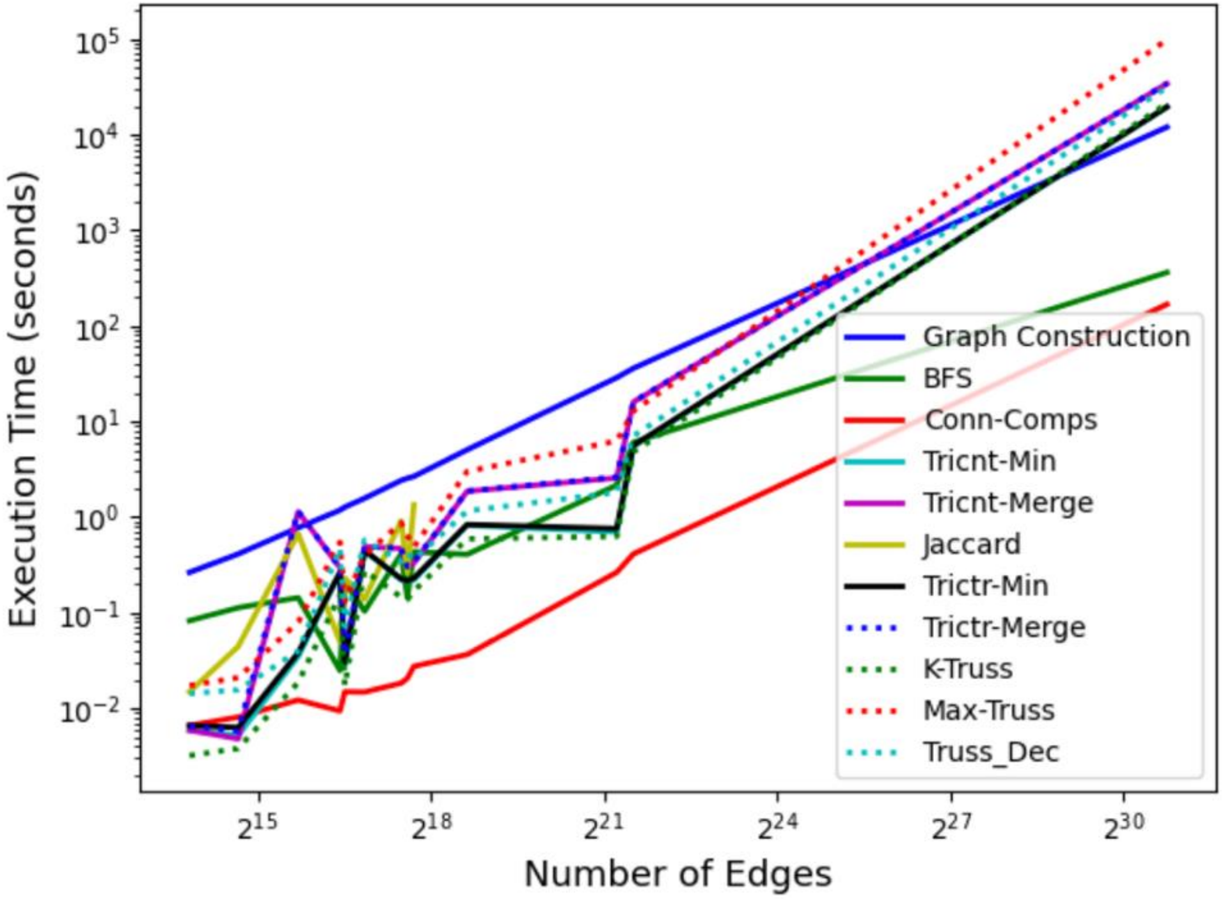
Synthetic

Graphs	Edges	Vertices	CCs	Triangles	Max K
delaunayn20	3145686	1048576	1	2109090	4
delaunayn21	6291408	2097152	1	4218386	4
delaunayn22	12582869	4194304	1	8436672	4
delaunayn23	25165784	8388608	1	16873359	4
delaunayn24	50331601	16777216	1	33746670	4

New Jersey Institute of Technology

Arachne Results – Real-World Graphs

[Alvarado Rodriguez, Du, Patchett, Li, Bader 2022]



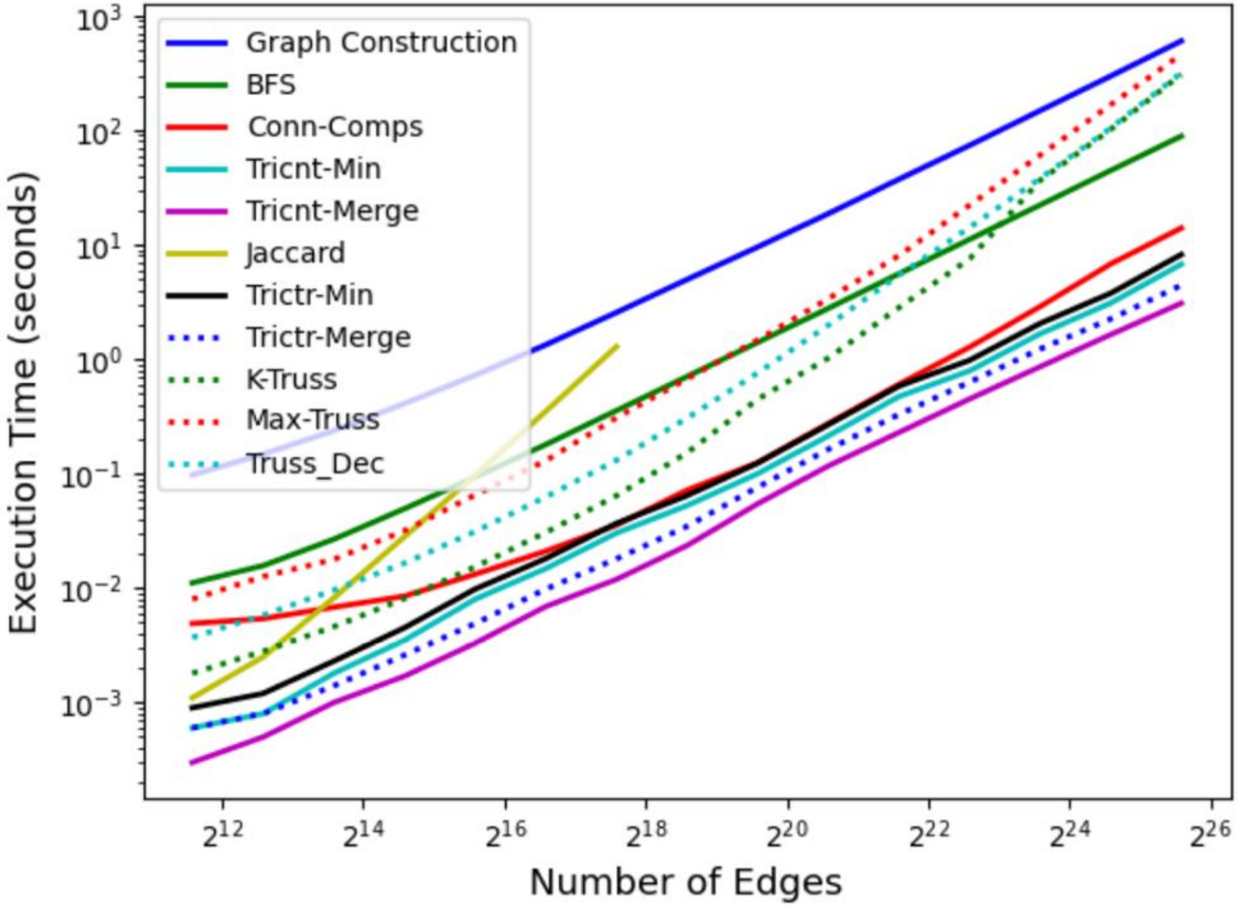
Key Points:

1. Graph construction is time consuming but once the graph is built into memory all the algorithms can use it in a highly efficient way.
2. The structural properties of graphs can significantly affect execution times even for the same algorithm.

New Jersey Institute of Technology

Arachne Results – Synthetic Graphs

[Alvarado Rodriguez, Du, Patchett, Li, Bader 2022]



Key Points:

1. Synthetic graphs demonstrate the scalability of our algorithms as the number of edges in a graph increase.
2. The memory requirements for each algorithm differ, hence the Jaccard coefficient algorithm encounters out of memory errors when the graph gets too big. Jaccard requires $\frac{N \times N}{2}$ memory and $\left(\frac{N}{P}\right)^2 \times \frac{M}{P}$ calculations.

New Jersey Institute of Technology

Breadth-First Search Improvements

Graph	num_vertices	num_edges	di (original)	di-norev	speedup
as-caida	26,475	53,381	2.22	1.22	1.82
delaunayn10	2,048	3,056	0.11	0.05	2.11
delaunayn20	1,058,576	3,145,686	90.49	47.61	1.90

Execution time in seconds on eight locales with 512GB memory and twenty processing units each.

```
while !SetCurF.isEmpty() do
  coforall loc in Locales do
    // parallel search on each locale
    forall i in SetCurF do
      if i is on current locale then
        SetNeighbor = ∪k, k is a neighbor of i
        forall j in SetNeighbor do
          if depth[j] == −1 then
            SetNextF.add(j)
            depth[j] = current_level + 1
          end
        end
      end
    end
  end
  SetCurF <=> SetNextF // exchange values
  SetNextF.clear()
  current_level+ = 1
end
```

Performed twice in di(original) while iterating over the edges in (SRC,DST) and in (SRCr, DSTr) to get the full neighborhoods which can lead to twice the number of remote reads and writes!

- About 50% improvement in number of PUTs and GETs with di-norev by including full neighborhoods of each vertex contiguously in one array instead of maintaining reversed edges.
- No change in storage volume, 2m edges still stored.
- Similar changes could optimize the rest of our graph kernels.

New Jersey Institute of Technology

STING Initiative: Focusing on Globally Significant Grand Challenges

- Many globally-significant grand challenges can be modeled by **Spatio-Temporal Interaction Networks and Graphs** (or "STING").
- Emerging real-world graph problems include:
  - Detecting community structure in large social networks
  - Defending the nation against cyber-based attacks
  - Discovering insider threats (e.g. Ft. Hood shooter, WikiLeaks)
  - Improving the resilience of the electric power grid
  - Detecting and preventing disease in human populations.
- Unlike traditional applications in computational science and engineering, solving these problems at scale often raises new research challenges due to:
  - Sparsity and the lack of locality in the massive data
  - Design of parallel algorithms for massive, streaming data analytics
  - The need for new exascale supercomputers that are energy-efficient, resilient, and easy-to-program

NJIT New Jersey Institute of Technology

STINGER – Time Frame

Pre-1999

Streaming graph need arises (over a decade ago)

2009

STINGER is officially proposed. May 2009

2010

First prototype, clustering coefficients. Apr 2010

2011

Structure tracking of streaming social networks. Apr 2011

2012

Dynamic betweenness centrality algorithm. Sep 2012

High Performance Data Structure for Streaming Graphs. Sep 2012
HPEC BEST PAPER AWARD

2013

Streaming connected component, Dec 2013

2014

Performance evaluation of open-source graph data-bases. Feb 2014

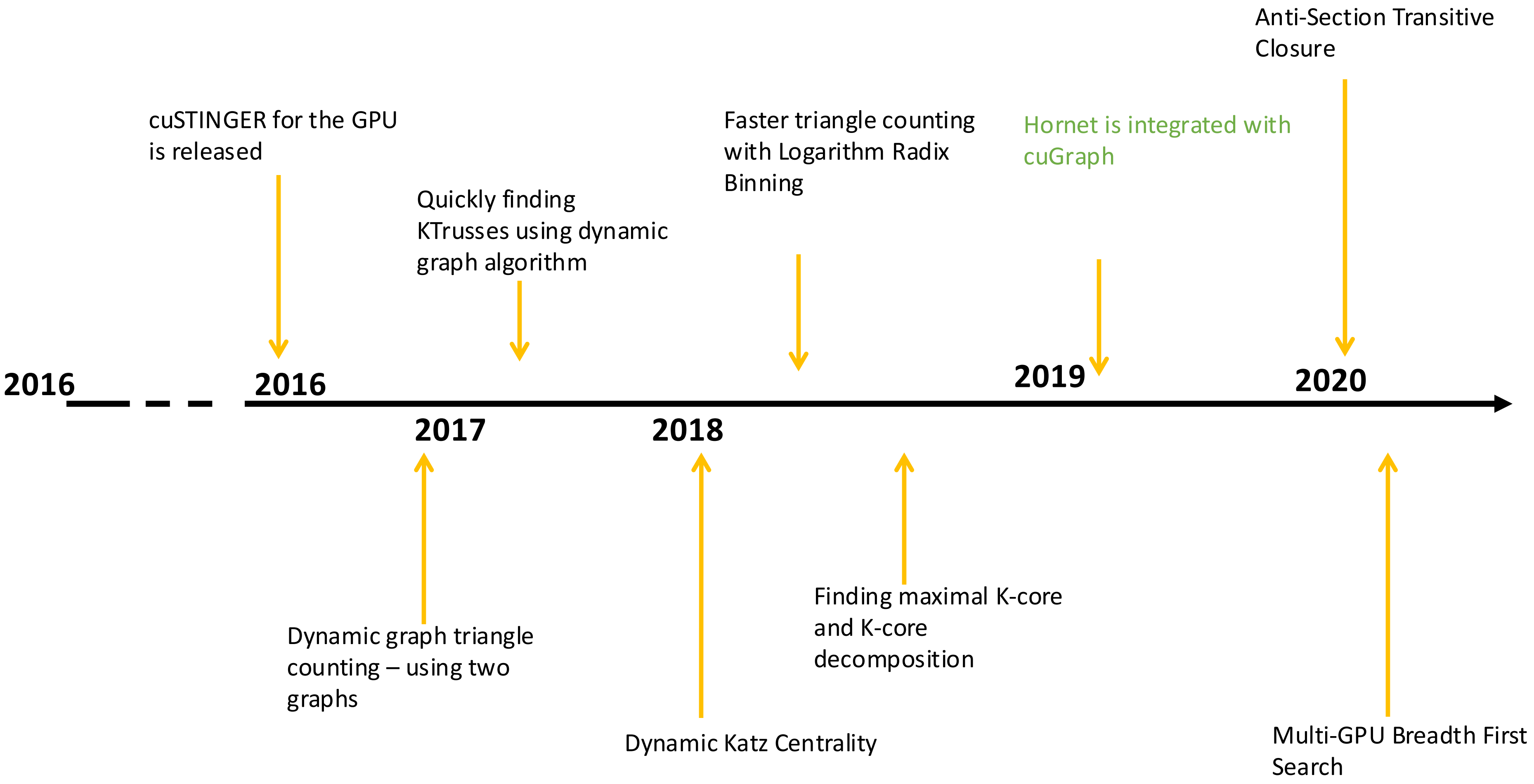
2015

Community detection in dynamic networks. Sep 2015

2016

PageRank for Streaming Graphs. May 2016

Hornet (GPU only) – Time Frame



New Jersey Institute of Technology

STING Extensible Representation (STINGER)

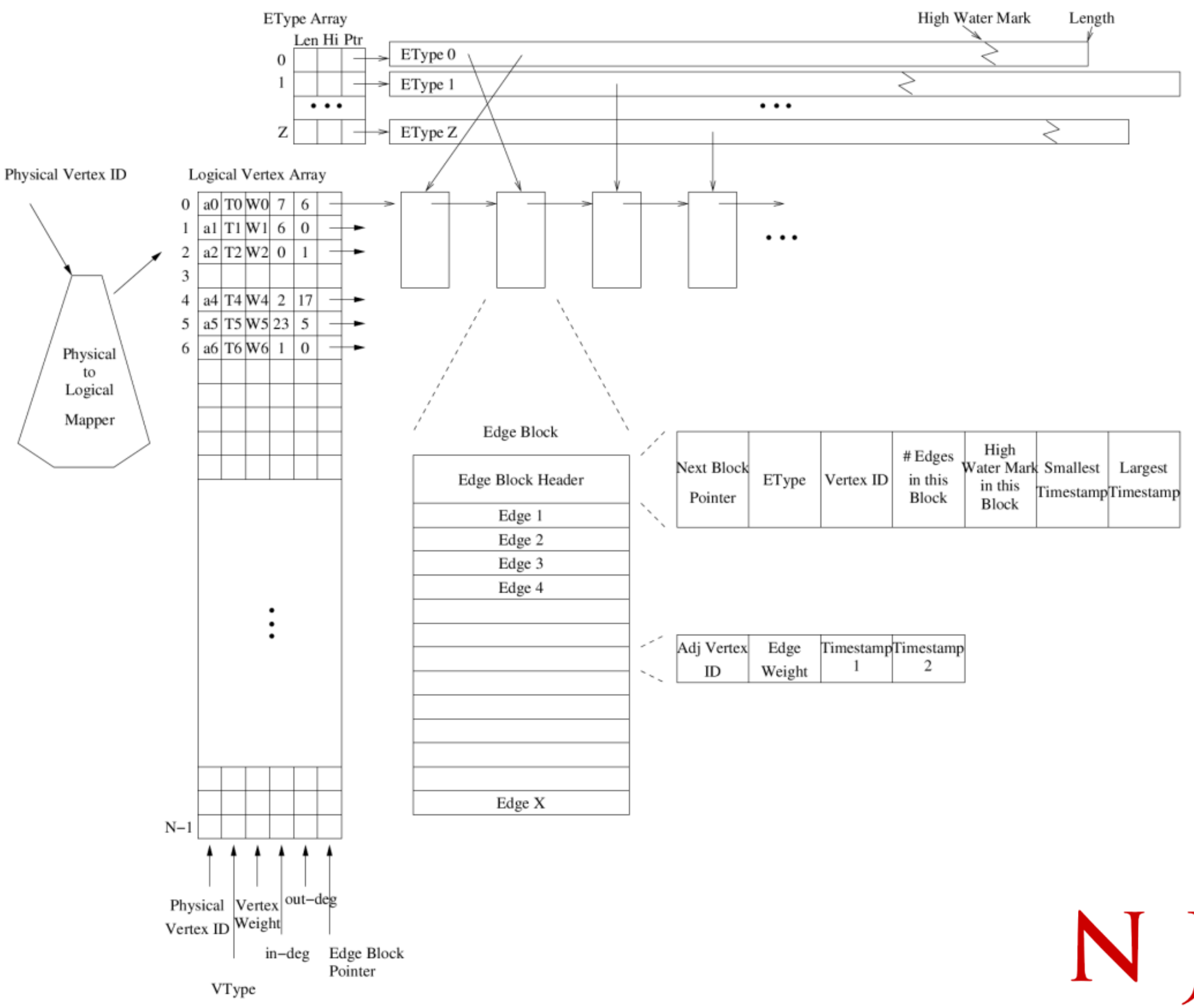
Design goals

- Enable algorithm designers to implement dynamic graph algorithms with ease.
- Portable semantics for various platforms
- Good performance for all types of graph problems and algorithms - static and dynamic.
- Assumes globally addressable memory access
- Support multiple, parallel readers and a single writer
  - One server manages the graph data structures
  - Multiple analytics run in background with read-only permissions.

New Jersey Institute of Technology

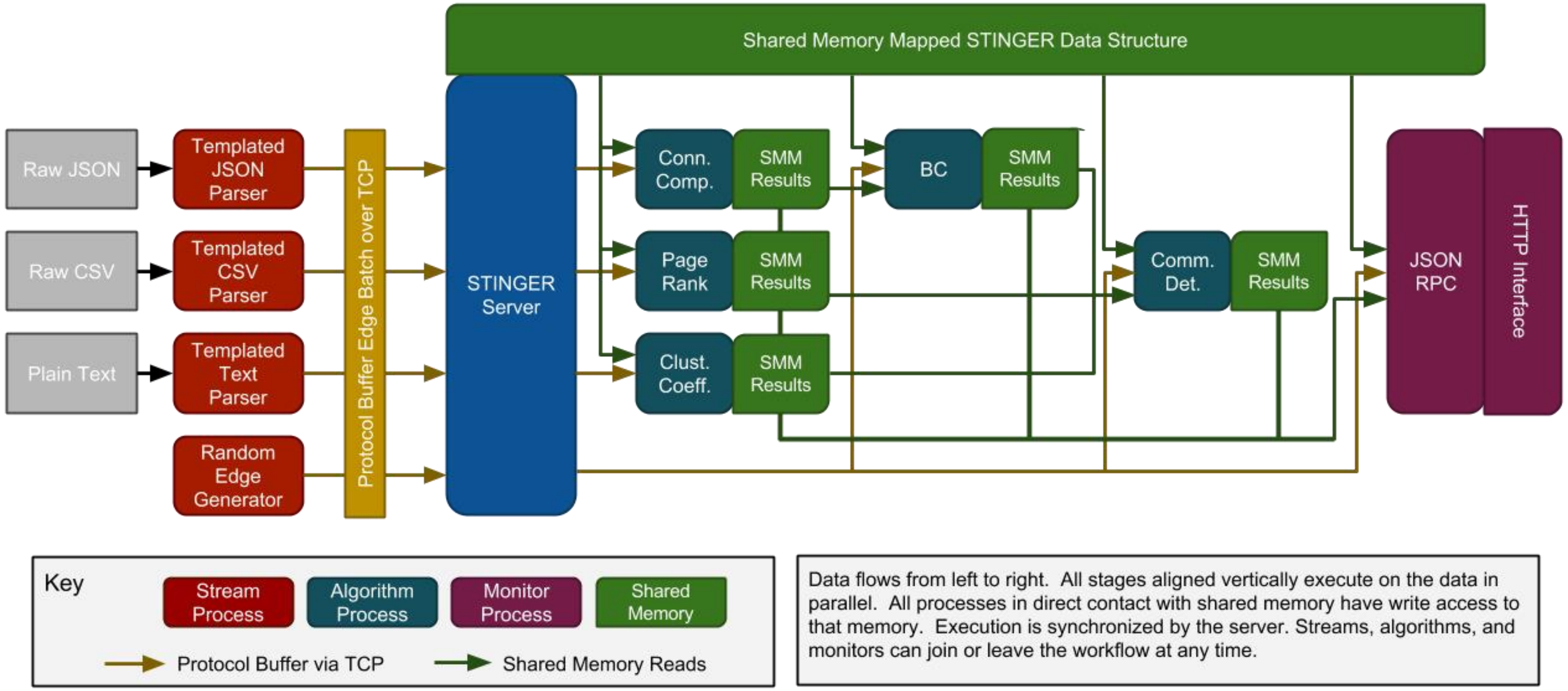
STING Extensible Representation (STINGER)

- Semi-dense edge list blocks with free space
- Compactly stores timestamps, types, weights
- Maps from application IDs to storage IDs
- Deletion by negating IDs, separate compaction



New Jersey Institute of Technology

STING: High-level architecture



- Server: Graph storage, kernel orchestration
- OpenMP + sufficiently POSIX-ish
- Multiple processes for resilience

New Jersey Institute of Technology

STINGER as an analysis package

http://www.stingergraph.com/

Anything that a static graph package can do (and a whole lot more):

Parallel agglomerative clustering:
Find clusters that are optimized for a user-defined edge scoring function.

K-core Extraction:
Extract additional communities and filter noisy high-degree vertices.

Classic breadth-first search:
Performs a parallel breadth-first search of the graph starting at a given source vertex to find shortest paths.

Parallel connected components:
Finds the connected components in a static network.

AND…

Streaming edge insertions and deletions:
New edge insertions, updates, and deletions in batches or individually. Optimized to update at rates of over 3 million edges per second on graphs of one billion edges.

Streaming clustering coefficients:
Tracks the local and global clustering coefficients of a graph.

Streaming connected components:
Real time tracking of the connected components.

Streaming Betweenness Centrality:
Find the key points within information flows and structural vulnerabilities.

Streaming community detection:
Track and update the community structures within the graph as they change.

NJIT New Jersey Institute of Technology

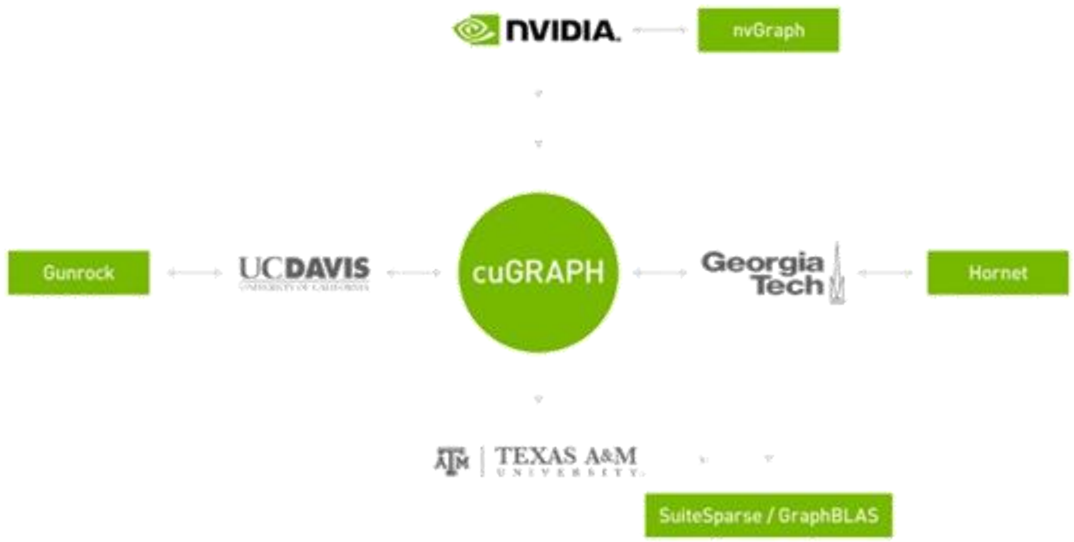
Why not existing technologies?

- Traditional SQL databases
  - Not structured to do any meaningful graph queries with any level of efficiency or timeliness
- Graph databases - mostly on-disk
  - Distributed disk can keep up with storing / indexing, but is simply too slow at random graph access to process on as the graph updates
- Hadoop and HDFS-based projects
  - Not really the right programming model for many structural queries over the entire graph, random access performance is poor
- Smaller graph libraries, processing tools
  - Can't scale, can't process dynamic graphs, frequently leads to impossible visualization attempts

NJIT
New Jersey Institute of Technology

AI Lab (NVAIL) 2019, PI: Bader
Building the Future of Graph Analytics with RAPIDS

"Prof. David Bader and his lab … are leaders in high performance computing algorithms, with a focus on both static and dynamic graph algorithms. With Prof. Bader and his lab, we will work on the design and implementation of scalable graph algorithms and graph primitives for integrating into cuGRAPH, leveraging their Hornet framework." – Sandra Skaff, NVIDIA, April 2019



NJIT New Jersey Institute of Technology

2019 Facebook AI Systems Award: Scalable Graph Learning Algorithms

Project Aim: Develop scalable graph learning algorithms and implementations that open the door for learned graph models on massive graphs

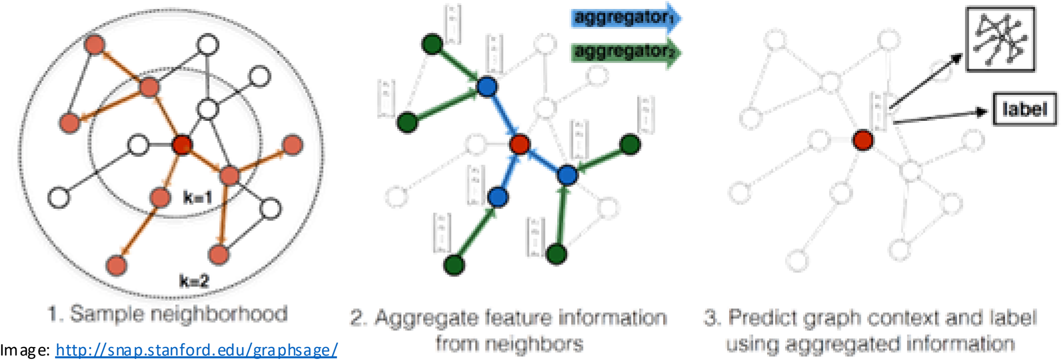


Image: http://snap.stanford.edu/graphsage/
Inductive Representation Learning on Large Graphs. W.L. Hamilton, R. Ying, and J. Leskovec arXiv:1706.02216 [cs.SI], 2017.

Deep Learning (DL) has significantly impacted the tasks of speech recognition, image classification, object detection and recommendation
Complex tasks: self-driving, super-human image recognition, recommendation engines, machine natural language translation, content selection, learning patterns of life
Techniques used in DL: convolutional neural networks (CNNs) → applicable for Euclidean data types and does not apply for Graphs
Solution: embedding graphs into a lower dimensional Euclidean space, generating a regular structure

1. developing a scalable high performance graph learning system based on GCNs algorithms, like GraphSage, by improving the workflow on shared-memory NUMA machines balancing computation between threads, optimizing data movement, and improving memory locality
2. investigate graph learning algorithm: specific decompositions and develop new strategies for graph learning that can inherently scale well while maintaining high accuracy

- Explore decomposition results from graph theory, for example forbidden graphs and the Embedding Lemma and determine how to apply such results into the field of graph learning
- Investigate whether these decompositions could assist in a dynamic graph setting

NJIT New Jersey Institute of Technology

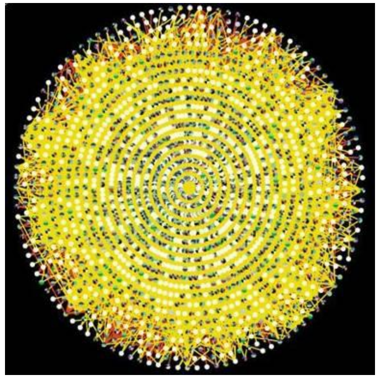
Graphs are pervasive in large-scale data analysis

- Sources of massive data: peta- and exa-scale simulations, experimental devices, the Internet, scientific applications.
- New challenges for analysis: data sizes, heterogeneity, uncertainty, data quality.

Astrophysics
Problem: Outlier detection.
Challenges: massive datasets, temporal variations.
Graph problems: clustering, matching.

Bioinformatics
Problem: Identifying drug target proteins.
Challenges: Data heterogeneity, quality.
Graph problems: centrality, clustering.

Social Informatics
Problem: Discover emergent communities, model spread of information.
Challenges: new analytics routines, uncertainty in data.
Graph problems: clustering, shortest paths, flows.



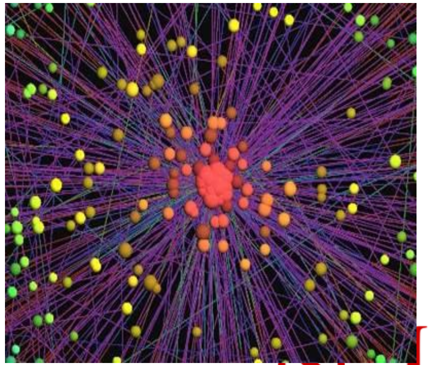


Image sources: (1) http://physics.nmt.edu/images/astro/hst_starfield.jpg
(2,3) www.visualComplexity.com

New Jersey Institute of Technology

Characterizing Graph-theoretic computations

Input: Graph abstraction

Problem: Find ***

- paths
- clusters
- partitions
- matchings
- patterns
- orderings

Graph algorithms

- traversal
- shortest path algorithms
- flow algorithms
- spanning tree algorithms
- topological sort

.....

Factors that influence choice of algorithm

- graph sparsity (m/n ratio)
- static/dynamic nature
- weighted/unweighted, weight distribution
- vertex degree distribution
- directed/undirected
- simple/multi/hyper graph
- problem size
- granularity of computation at nodes/edges
- domain-specific characteristics

Graph problems are often recast as sparse linear algebra (e.g., partitioning) or linear programming (e.g., matching) computations

NJIT New Jersey Institute of Technology

Streaming Analytics move us from reporting the news to predictive analytics

Traditional HPC

- Great for “static” data sets.
- Massive scalability at the cost of programmability.
- Great for dense problems.
  - Sparse problems typically underutilize the system.

Streaming Analytics

- Requires specialized analytics and data structures.
- Rapidly changing data.
- Low data re-usage.
  - Focused on memory operations and not FLOPS.

NJIT
New Jersey Institute of Technology

Graph Data Science

- Are there new graph techniques? Do they scale? Can the computational systems (algorithms, machines) handle massive networks with billions to trillions of items? Can the techniques tolerate noisy data, massive data, streaming data, etc. …
- **Communities may overlap, exhibit different properties and sizes, and be driven by different models**
  - **Detect communities (static or emerging)**
  - **Identify important individuals**
  - **Detect anomalous behavior**
  - **Given a community, find a representative member of the community**
  - **Given a set of individuals, find the best community that includes them**
  - **Find congestion, weak points, anomalies, surprises, …**

New Jersey Institute of Technology

Massive Streaming Graph Analytics

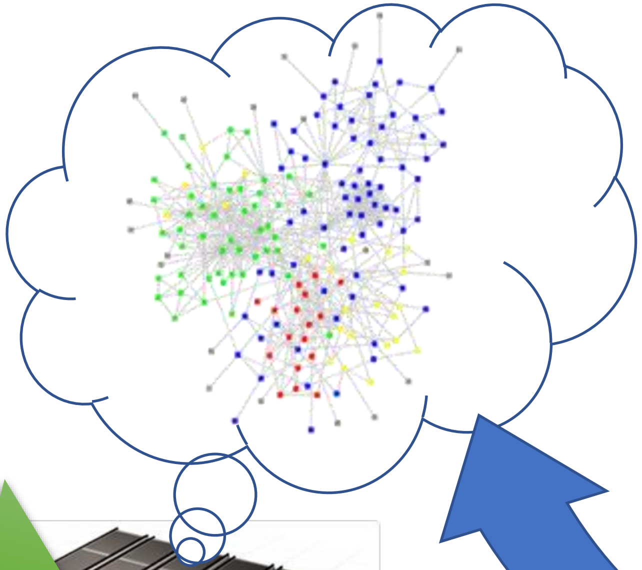
(A, B, t1, poke)
(A, C, t2, msg)
(A, D, t3, view wall)
(A, D, t4, post)

(B, A, t2, poke)
(B, A, t3, view wall)
(B, A, t4, msg)

Billions of edges
… e9 e8 e7 e6 e5 e4 e3 e2 e1 …

Analysts

Q3? Q2? Q1?



NJIT
New Jersey Institute of Technology

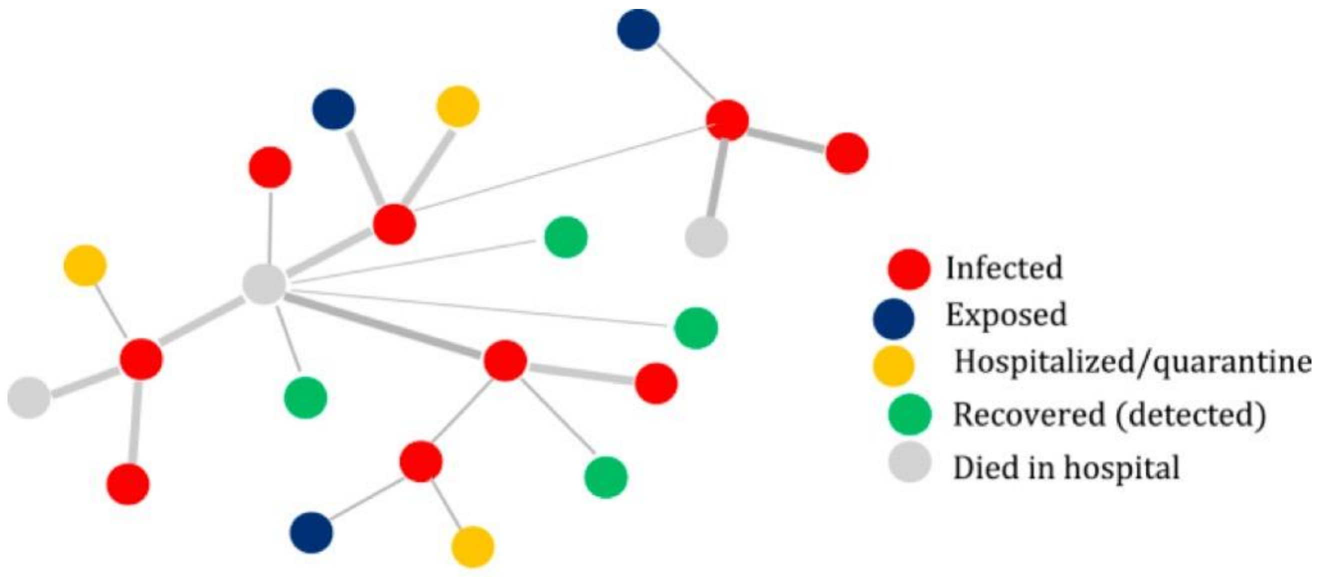
Hierarchy of Interesting Analytics

- **Extend single-shot graph queries to include time.**
  - Are there *s*-*t* paths between time T_1 and T_2?
  - What are the important vertices at time *T*?
- **Use persistent queries to monitor properties.**
  - Does the path between *s* and *t* shorten drastically?
  - Is some vertex suddenly very central?
- **Extend persistent queries to fully dynamic properties.**
  - Does a small community stay independent rather than merge with larger groups?
  - When does a vertex jump between communities?
- **New types of queries, new challenges...**

New Jersey Institute of Technology

Modeling Pandemic Spread

- The graph represents the contact patterns between individuals in a population.
- Various graph algorithms can be used to simulate the spread of a pandemic.
  - Centrality measures such as eigenvector centrality can identify the most important vertices in the network
  - Visualization of the spread of the pandemic can be created to check the effects of intervention and control strategies.
- The dataset can be a million or even a trillion vertices.



[Alguliyev, Aliguliyev, Yusifov, 2020]

R. Alguliyev, R. Alguliyev, and F. Yusifov, "Graph modelling for tracking the COVID-19 pandemic spread." Infectious disease modelling, 6, 2021: 112-122

NJIT New Jersey Institute of Technology

Using Arachne with Arkouda (1/3)

In [2]:
```
ak.connect("d-6-15-4", 5555)
```

```
connected to arkouda server tcp://*:5555
```

In [3]:
```
# Read in the graph.
filename = "/home/gridsan/oarodriguez/biggraph_shared/Adata/simple.txt"
ne = 13
nv = 10
G = ar.graph_file_read(ne, nv, 2, 0, filename, 1, 0, 0, 0, 1)
```

```
13 10 2 0 /home/gridsan/oarodriguez/biggraph_shared/Adata/simple.txt 1 0 0 0 1
```

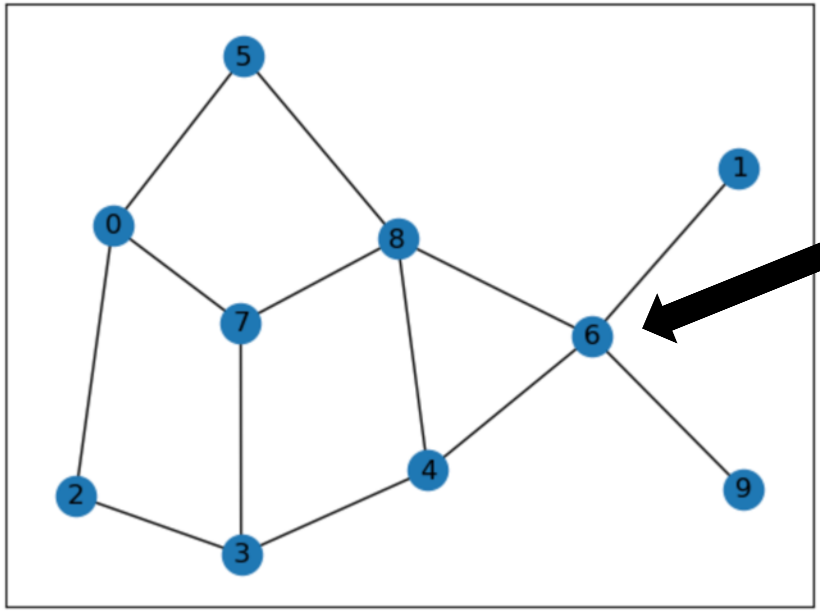
In [4]:
```
# Add the edges of the graph to a list of tuples.
src = ar.graph_query(G, "src")
dst = ar.graph_query(G, "dst")

edges = []
for (u, v) in zip(src.to_ndarray(), dst.to_ndarray()):
    edges.append((u,v))
```

In [5]:
```
# Display the graph with NetworkX.
nxG = nx.Graph()
nxG.add_edges_from(edges)

pos = nx.spring_layout(nxG, seed=225)
nx.draw_networkx(nxG, pos, with_labels=True)
plt.show()
```

Using Arachne with Arkouda (2/3)



max degree! (also 8, but 6 is the first occurrence)

In [6]:
```
# Get value of the maximum degree.
neighbour = ar.graph_query(G, "neighbour")
neighbourR = ar.graph_query(G, "neighbourR")
degrees = neighbour + neighbourR
print("The value of the maximum degree is: {}".format(ak.max(degrees)))
```

```
The value of the maximum degree is: 4
```

NJIT New Jersey Institute of Technology

Using Arachne with Arkouda (3/3)

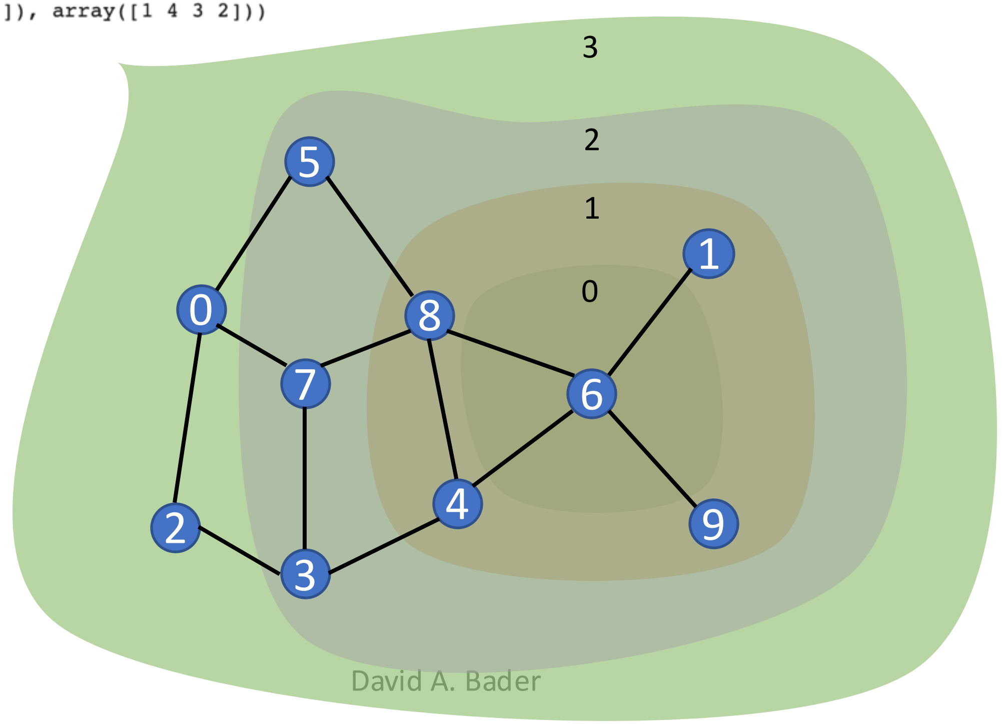
Breadth-First Search

In [7]:
```
d = ar.graph_bfs(G, int(ak.argmax(degrees)), 0)
print(d)
```

```
[3 1 3 2 1 2 0 2 1 1]
```

In [8]:
```
# Get the size of each level of BFS.
d_histogram = ak.histogram(d, bins=ak.max(d)+1)
print(d_histogram)
```

```
(array([0.  , 0.75, 1.5 , 2.25]), array([1 4 3 2]))
```



New Jersey Institute of Technology