Massive Dataset Analysis in Arkouda

David A. Bader

@Prof_DavidBader

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

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 \$186M of research awards
- 300+ publications, ≥ 13,000 citations, h-index ≥ 65
- 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.

NJIT
New Jersey Institute of Technology

Dedication:
Morris Bader
(26 October 1932 – 21 April 2005)

Morris Bader, 72, of Bethlehem, died peacefully at home on April 21, 2005. He was the husband of Karen (Roberts) Bader. They were married for 45 years. Born in New York, he was the son of the late Louis and Esther (Saltzman) Bader. He was a graduate of Stuyvesant High School in New York City. He was a 1953 graduate of the City University of New York (formerly City College of New York) and earned his Ph.D. in physical chemistry at Indiana University, Bloomington, IN. He taught at New York University, Marietta College in Marietta, OH, and Moravian College. He was an emeritus professor of chemistry at Moravian College. He taught chemistry and computer science from 1962 until his retirement in 1995. He also taught physical chemistry, **developed the initial computer science program**, conceived and funded the SOAR program for funding student and faculty summer research, and collaborated and developed a plant growth hormone. He was a scientific glassblower, making much of his own equipment. **He developed scientific programs and published five computer manuals and software which sold worldwide, the profits of which were donated to assist faculty research travel to conferences**. He developed the course "Chemistry for the Non-Science Major" and his paper "A Systematic Approach to Standard Addition Methods in Instrumental Analysis" is highly-cited and used widely in practice. He holds two patents: one for a bicycle gearing system and one for a quartz infrared cell; both manufactured. **He has published numerous articles in numerical scientific computation for chemical analysis, solution of hard differential equations, and improved accuracy and error analysis in numerical computing**. His chemistry publications include various chemical experiments for use by educators, and guidelines and error estimates for the neglect of buoyancy in laboratory weighings. He has published in the Journal of Chemical Education and in American Laboratory of which he was a contributing editor. He was a championship chess player and the Moravian College Chess Club advisor. He supported the Moravian College Foreign Film Festival. His many volunteering activities included teaching swimming to toddlers at the 3rd St. Alliance (Easton), Rodale Theater, State Theatre (Easton), 21-year Musikfest volunteer, LVH-Muhlenberg Hospital, Financial Advisor to the Friendship Circle of the J.C.C., and Assistant Scoutmaster of Troops 304 and 346 in the Minsi Trails Council, Boy Scouts of America. Morris was a member of Congregation Beth Avraham (formerly Agudath Achim) of Bethlehem; he was a Member of the Board and then President for 20 years. He read Torah and led services.

SURVIVORS: Wife; Sons, William A. of Bethlehem, Joel S. and his wife Jennifer of Baltimore, MD, David A. and his wife Sara Gottlieb of Albuquerque, NM; daughter, Debra S. and her husband Eric Eisenstein of Ithaca, NY; sisters: Rose and husband Ralph Hittman, Marion Ashrey, all of New York City; and five grandchildren.

SERVICES: graveside, were held Friday, April 22, Beth Avraham / Agudath Achim Cemetery, Fountain Hill. Arr. by Long Funeral Home, Bethlehem.

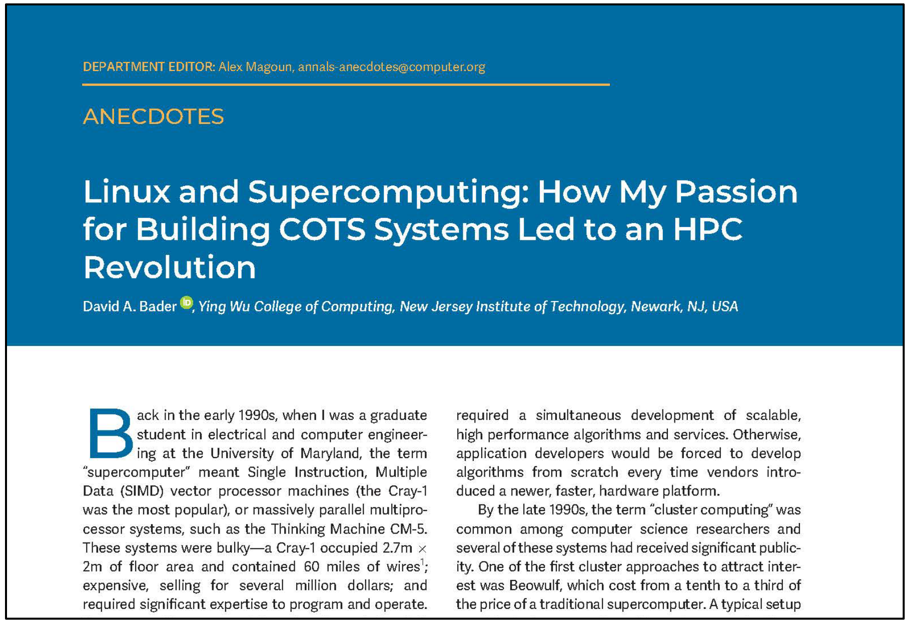
MEMORIALS: in Morris's memory made be made to Congregation Beth Avraham, 1555 Linwood Street, Bethlehem, PA, 18017; or to the Bader Memorial Scholarship Prize in Chemistry, Moravian College, 1200 Main St., Bethlehem, PA, 18018.

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

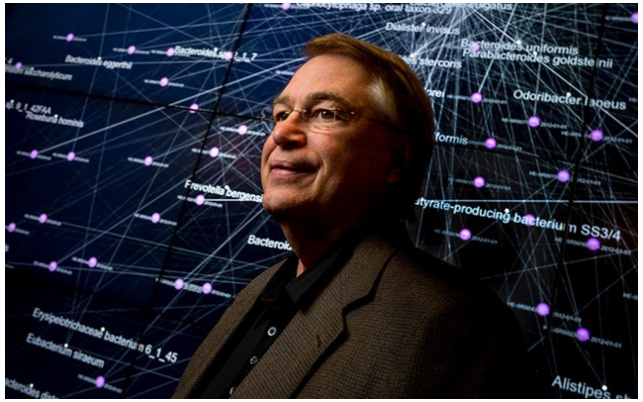
ANECDOTES

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

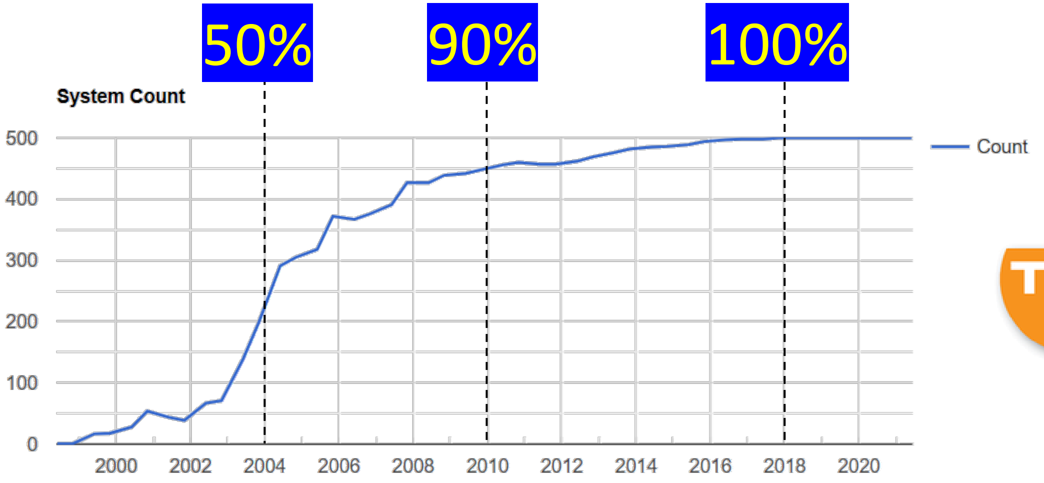
NJIT New Jersey Institute of Technology

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

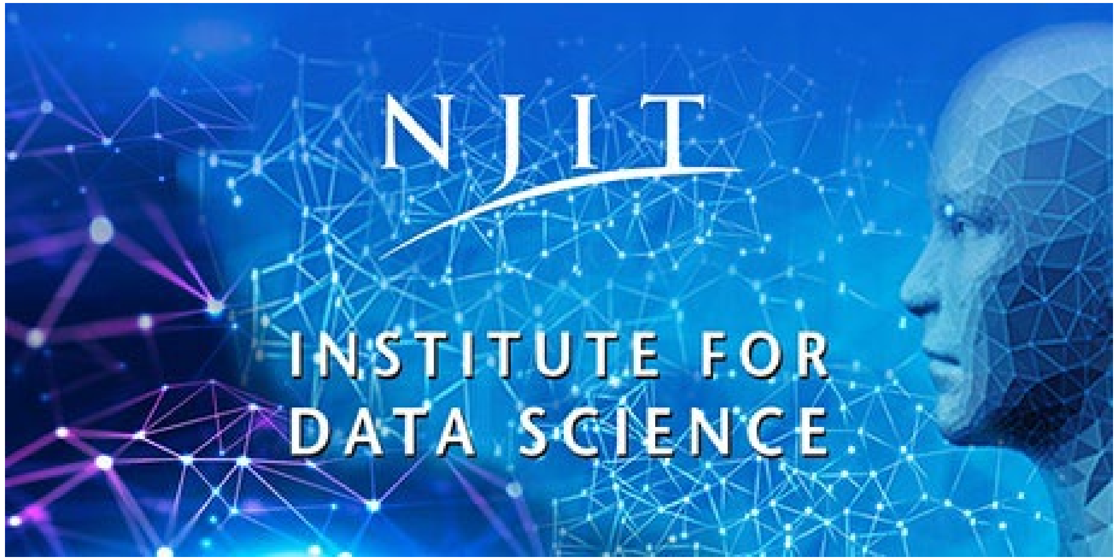
New Jersey Institute of Technology

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

NJIT
New Jersey Institute of Technology



Launched in **July 2019**, with inaugural director
David A. Bader
(~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

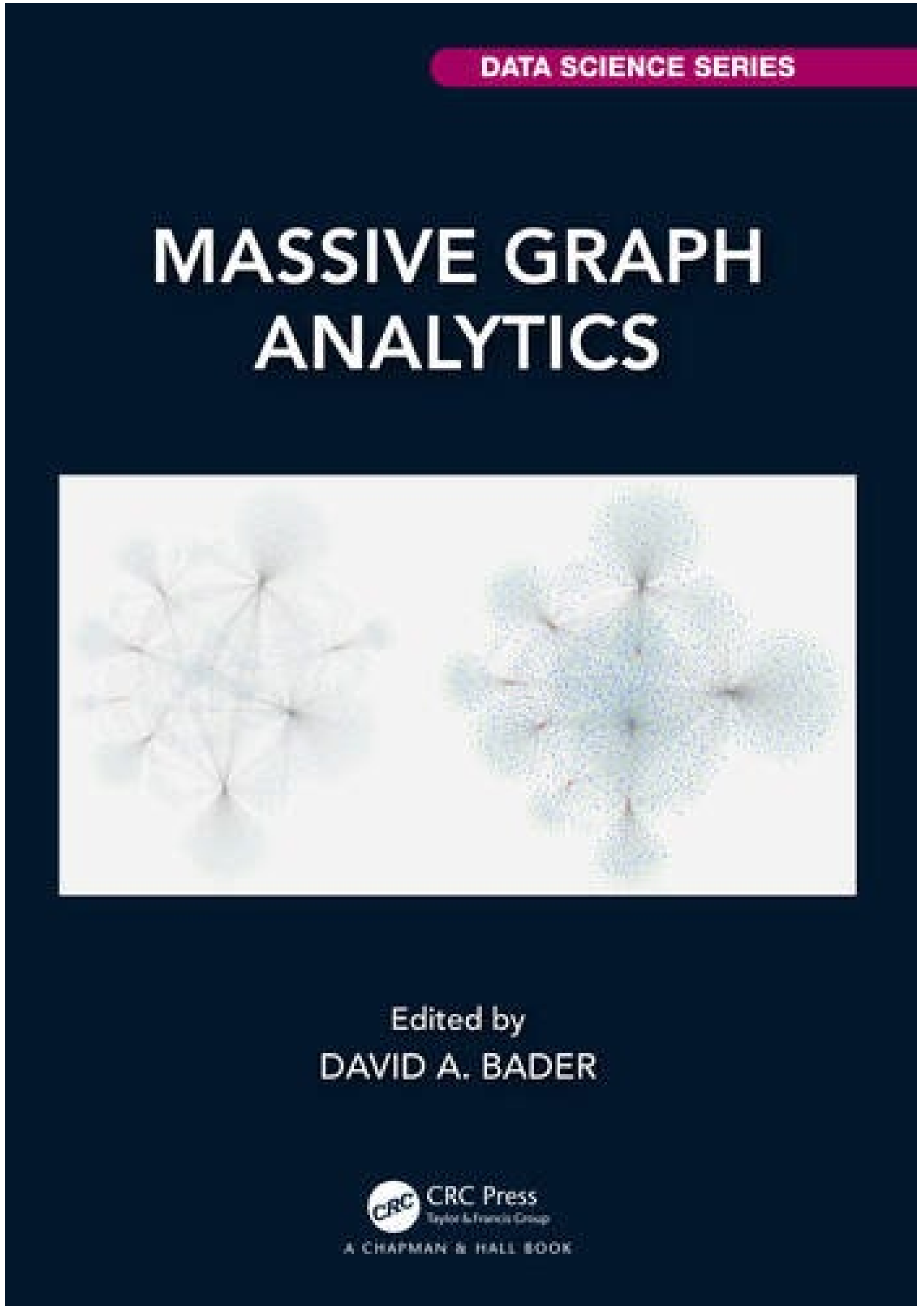
Cybersecurity Research Center
- Practical encryption
- Privacy technologies
- Information Assurance

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

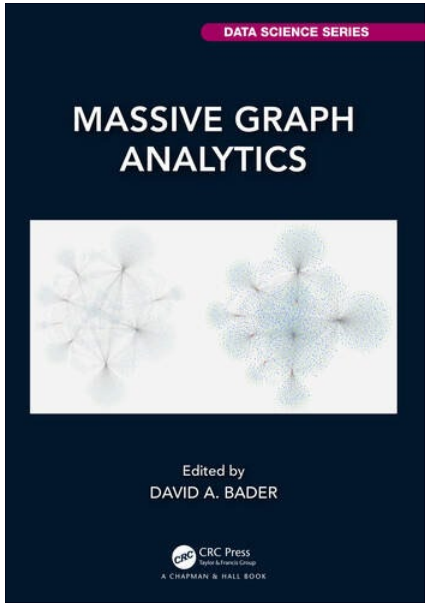
NJIT
New Jersey Institute of Technology



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

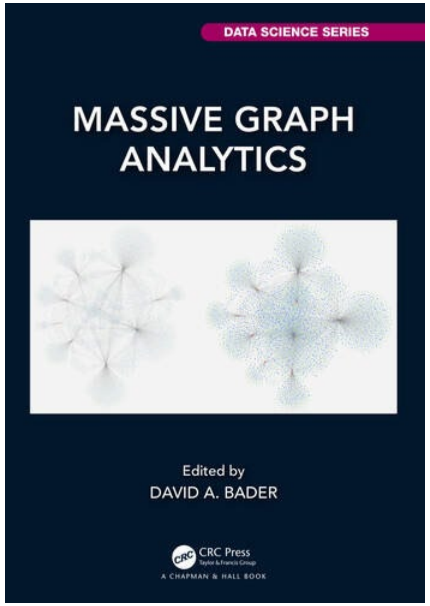
A Cloud-based approach to Big Graphs
Paul Burkhardt and Christopher A. Waring

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

Edge Computing

- Bring computations closer to the devices that are performing the data collection instead of transmitting it straight to the cloud.
- Consists of decentralized architectures with a high need for low latency and network connections to other edge nodes.

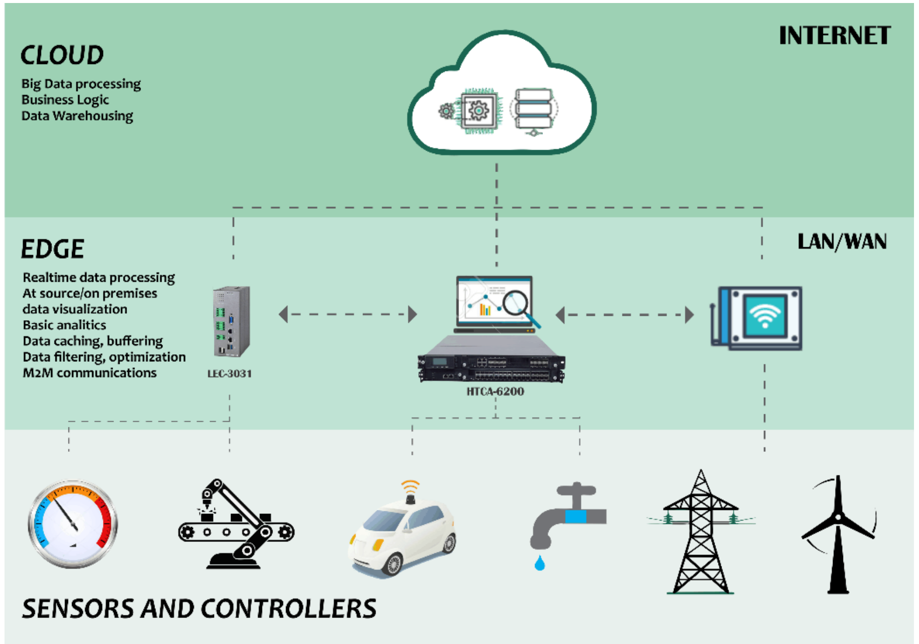
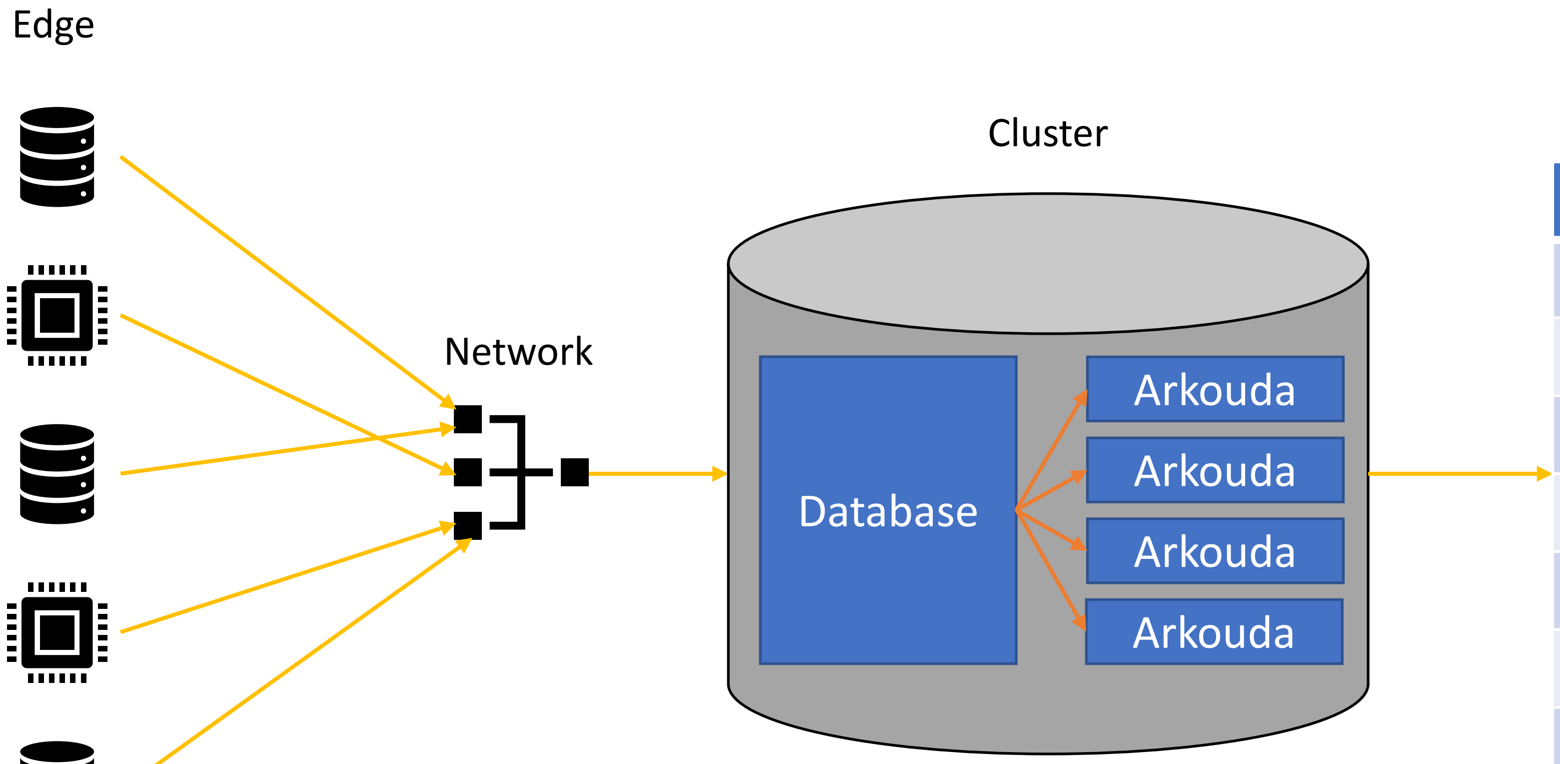


Image source: https://www.lanner-america.com/blog/4-edge-computing-technologies-enabling-iot-ready-network-infrastructure-2018/

To the Edge…



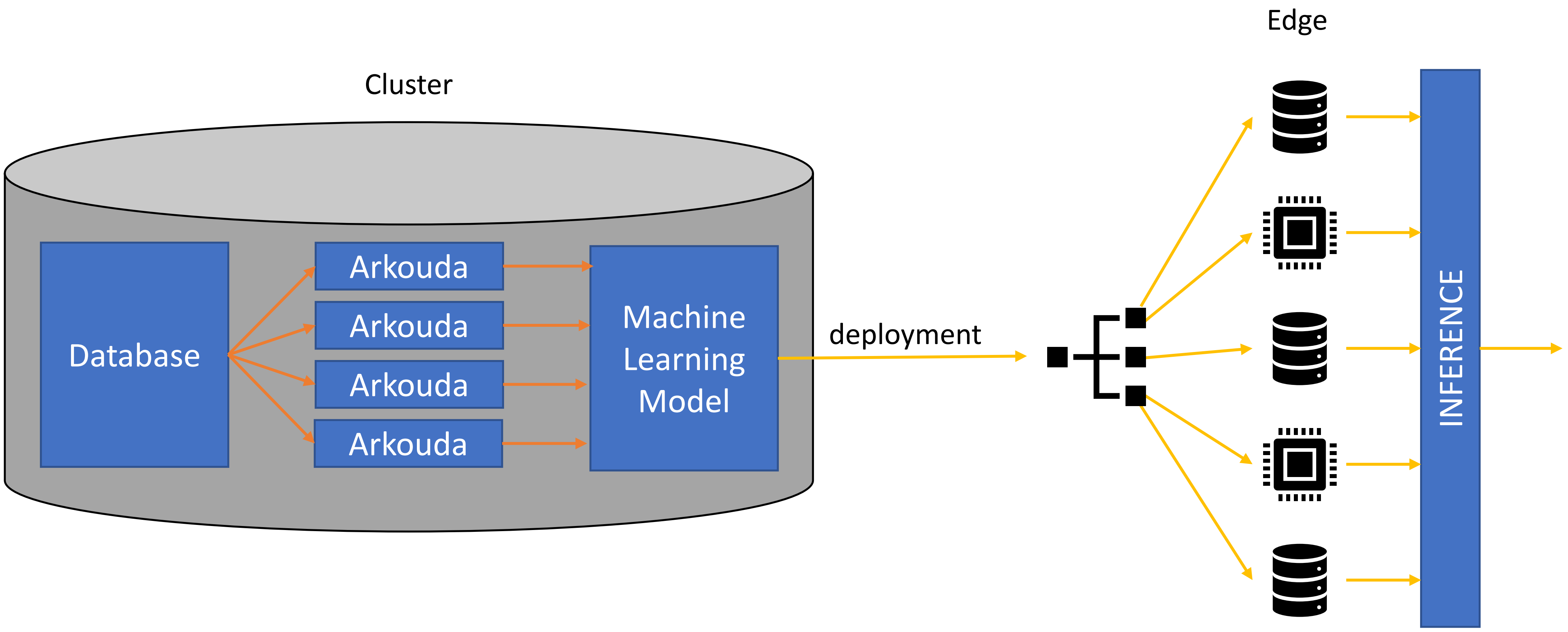
Time	Data
0:00	jdkbfjksdbfbfasdkfb
0:15	?????????????????
0:30	sdfjhsdjkfhjskalhdfjk
0:45	asjdhfkasdhfkasdhfkj
1:00	asdfhjasdhfsadhfahs
1:15	??????????????????
1:30	djfhjsdkfhkjashdfjkha

Data transferred from edge to cluster.

Data processed on multilocale Arkouda server.

Revealed unanticipated periodicity in the data!

And back.


Edge
Cluster
Database
Arkouda
Arkouda
Arkouda
Arkouda
Machine Learning Model
deployment
INFERENCE

NJIT 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

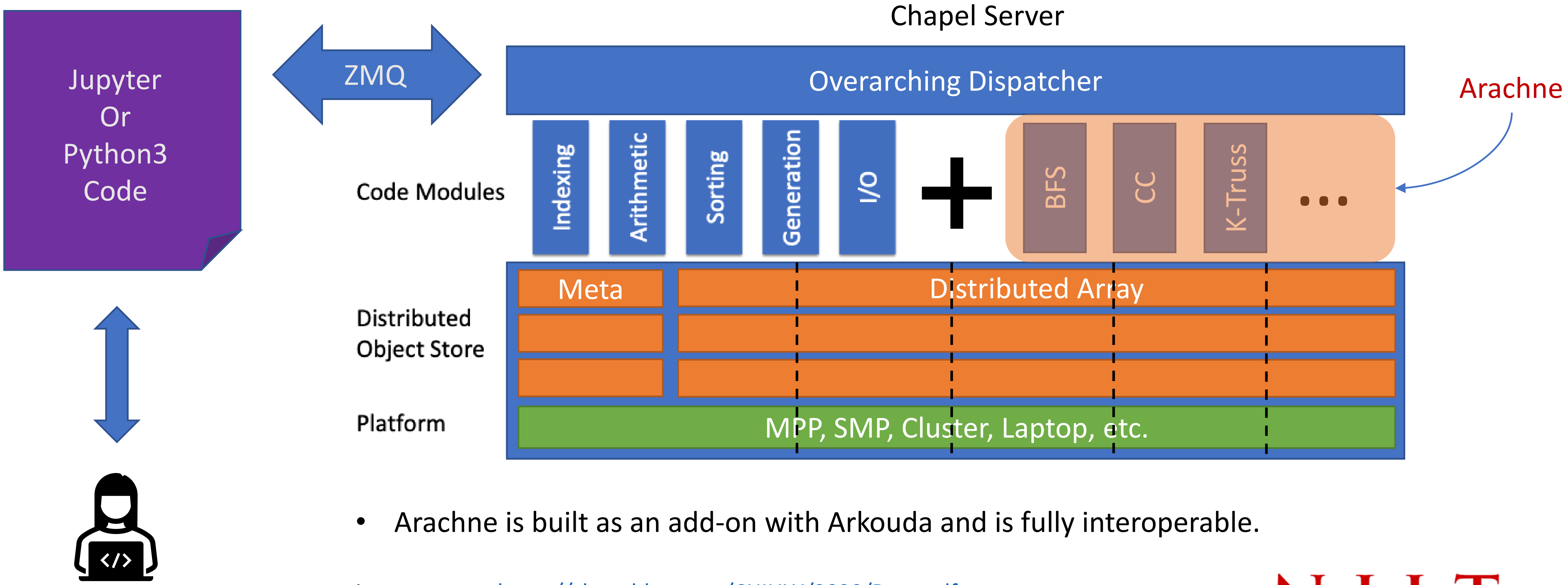
(PI: Bader)
3/2021 – 2/2023
NSF CCF-2109988

αρκούδα
massive scale data science

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

NJIT New Jersey Institute of Technology

The Arkouda Framework with Arachne



- Arachne is built as an add-on with Arkouda and is fully interoperable.

Image source: https://chapel-lang.org/CHIUW/2020/Reus.pdf

NJIT
New Jersey Institute of Technology

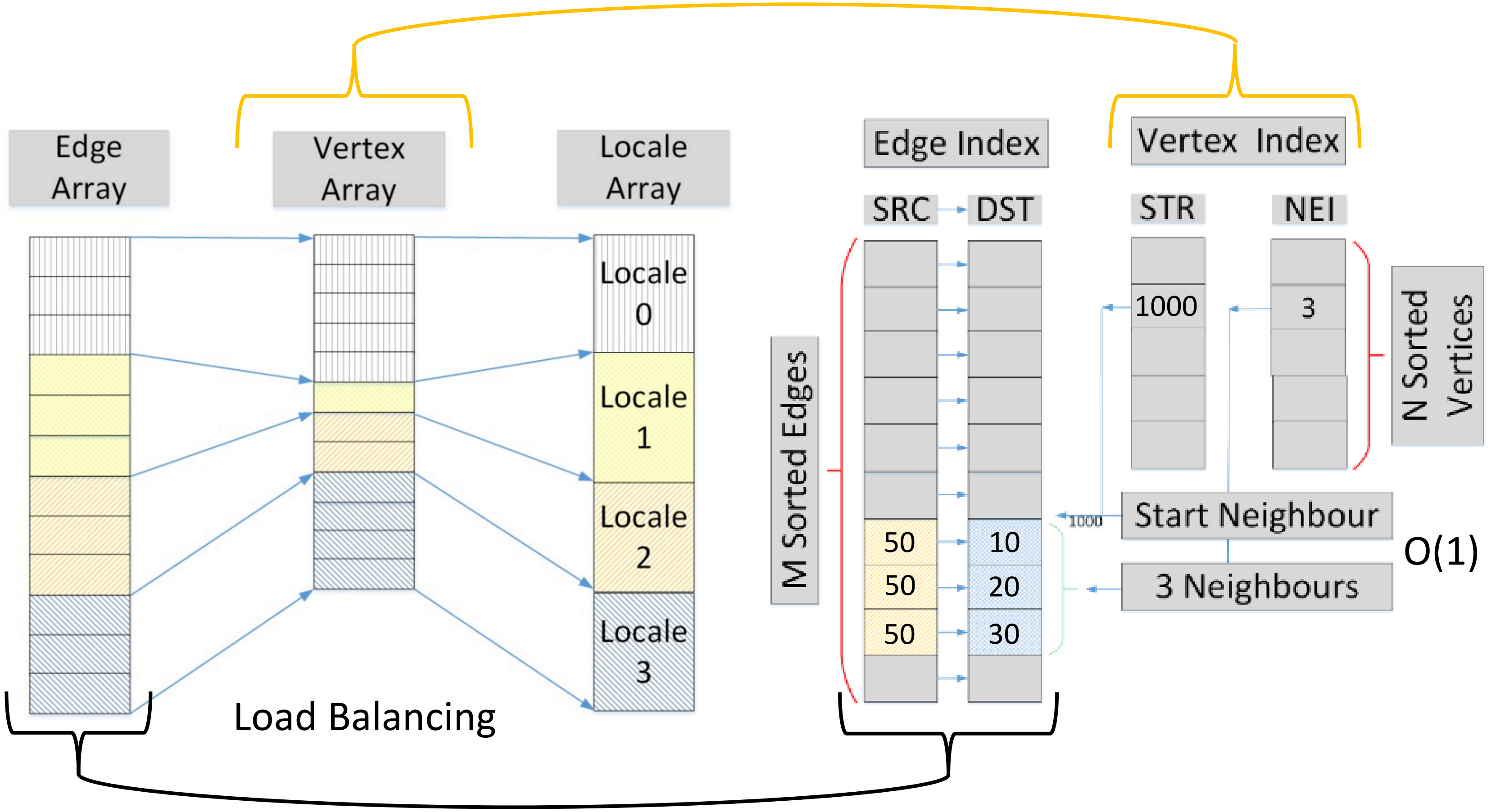
Major Contributions

- 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

Arachne Double-Index (DI) Data Structure

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

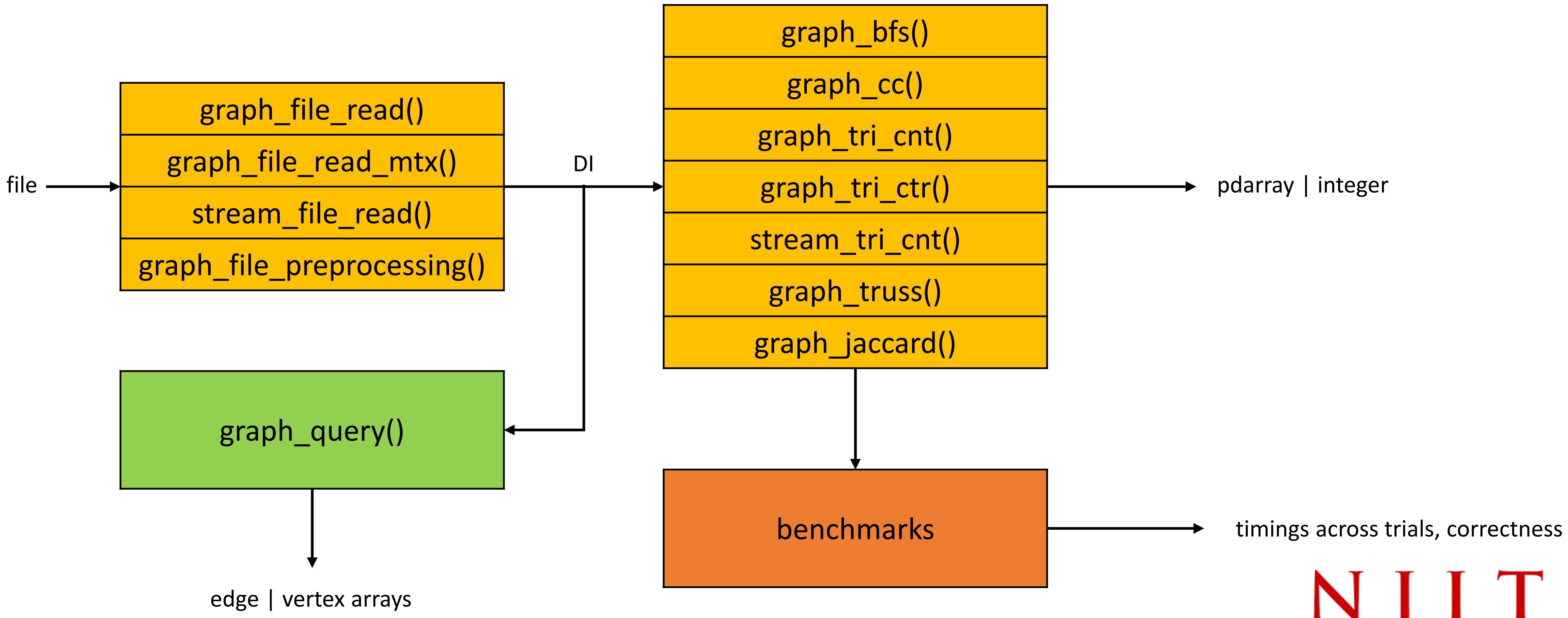


Advantages of DI over CSR:

1. O(1) time complexity:
   - locating a vertex from a given edge ID.
   - locating the adjacency list from a given vertex ID.
2. We can search from edge ID to vertex ID, this is not possible in CSR.
3. DI can support both edge-centric and vertex-centric algorithms whereas CSR can only support the latter.
4. DI can easily achieve load balancing with the edge array being distributed equally amongst many locales.

NJIT New Jersey Institute of Technology

Modules of Arachne



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.
- **Truss Analytics** [Patchett, Du, Bader 2021]
  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.
- **Jaccard Coefficients**
  Returns an array of size n with the Jaccard metric between some vertices u and v.
- **Triangle counting** [Du, Alvarado Rodriguez, Patchett, Bader 2021]
  Returns the number of triangles in a graph.
- **Triangle centrality** [Patchett, Du, Bader 2022]
  Returns an array of size n with the proportion of triangles centered at a vertex v.

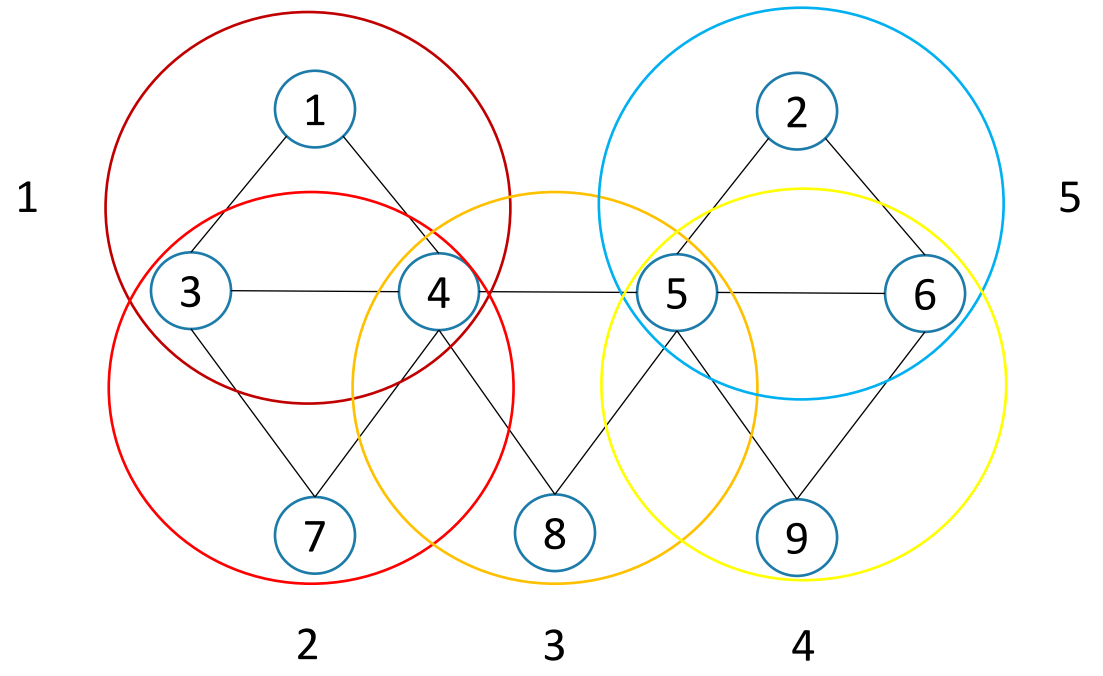
NJIT
New Jersey Institute of Technology

Algorithmic Contributions

Algorithm	Novelty
Breadth-First Search	High-level algorithm based off Chapel's high-level distributed bag data structure outperforms low-level manual aggregation algorithm.
Connected Components	Fast-spreading algorithm that involves propagating the lowest vertex label of each component to all other vertices. It completes in $O(\log(d_{max}) + 1)$ iterations where d_{max} is the diameter for the largest connected component.
Triangle Counting	Minimum search triangle counting can identify two smaller adjacency lists and employ fine-grained parallelism to achieve better performance by searching for a vertex to complete a triangle edge in the smaller adjacency list.
Triangle Centrality	Uses minimum search triangle counting to find (1) the sum of the number of triangles of the given vertex's adjacency list and itself, and (2) the sum of the number of triangles of the given vertex's neighbors that are connected by a triangle.
Truss Analytics	Uses minimum search triangle counting to speed up the triangle search process of truss edges.
Jaccard Coefficients	Edge-centric graph partitioning allows for high memory access locality.

What is Triangle Counting?

Input:



Triangle_count = 5

NJIT New Jersey Institute of Technology

Minimum Search Triangle Counting (1/3)

[Du, Alvarado Rodriguez, Patchett, Bader 2021]

Algorithm 3: *Parallel Minimum Search* based Triangle Counting

```
Input: A graph G = (V, E).
Output: An integer value of the number of triangles.
coforall loc in Locales do
    forall (edge e = u,v ∈ E) && (e is local) do
        // We assume that |Adj(u)|<|Adj(v)|
        var count:int=0;
        forall w ∈ Adj(u) with (+ reduce Count ) do
            if (|Adj(w)| < |Adj(v)|) then
                if (v ∈ Adj(w)) then
                    count++;
                end
            end
            else
                if (w ∈ Adj(v)) then
                    count++;
                end
            end
        end
    end
end
return count
```

get smallest adjacency list

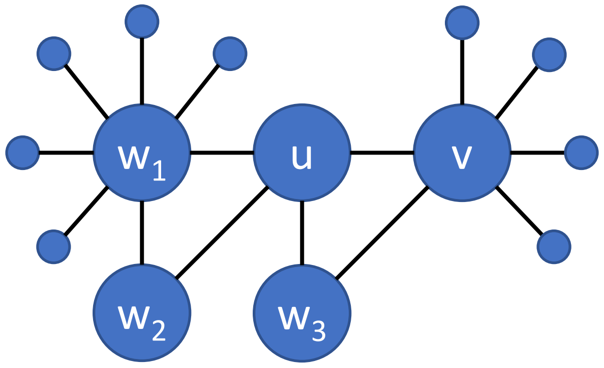
get closing triangle edge

Total time to search all triangles with a given edge: $\max_{w \in Adj_l} \log_2(\min(|Adj_w|, |Adj_h|))$ vs. list intersection $\log_2(|Adj_h|)$ where *h* is *u* or *v* with more adjacent vertices, *l* with less.

Minimum Search Triangle Counting (2/3)

[Du, Alvarado Rodriguez, Patchett, Bader 2021]

1. Given an edge (u, v) we assume that $|Adj(u)| \leq |Adj(v)|$.
2. Then, for $\forall w \in Adj(u)$ we spawn $|Adj(u)|$ 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.

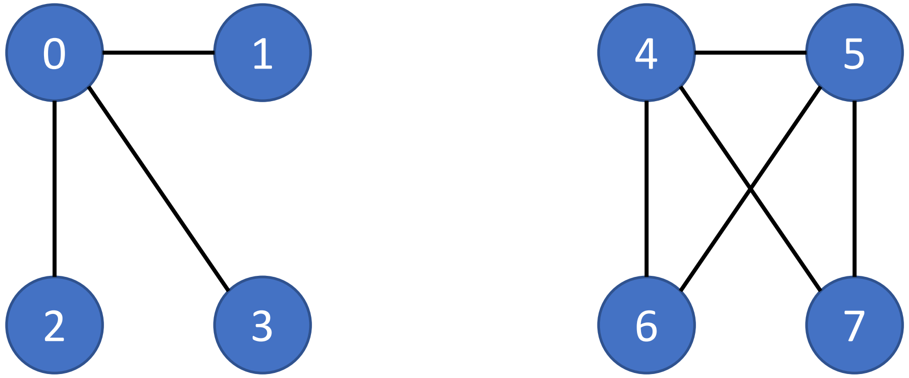
NJIT
New Jersey Institute of Technology

Minimum Search Triangle Counting (3/3)
[Du, Alvarado Rodriguez, Patchett, Bader 2021]

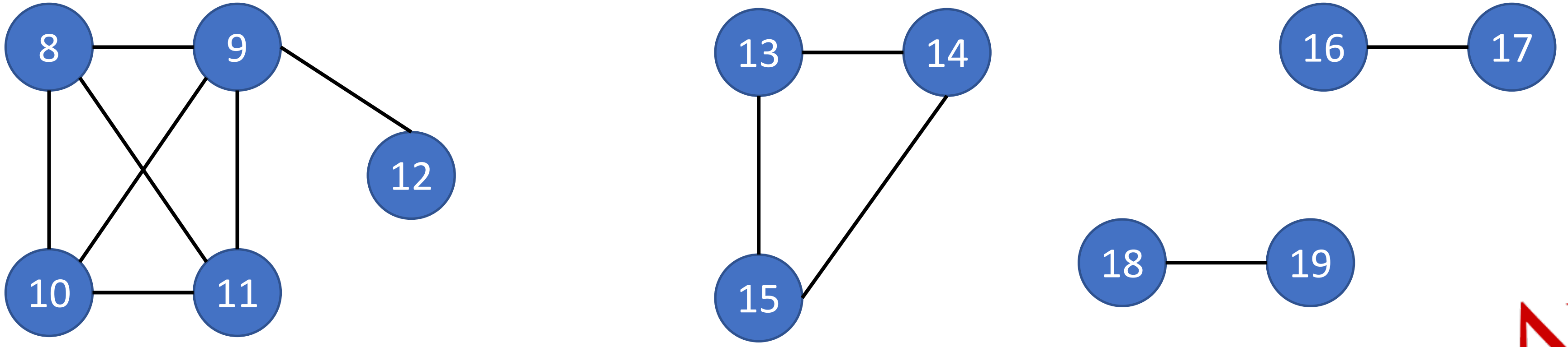
- Why does our minimum search method have less operations?
- If $|Adj_l| = 4$, and for every w in Adj_l, $|Adj_w| \leq 8$, and $|Adj_h| = 1024$ then list intersection will use four parallel threads that amount to $\lceil \log_2 1024 \rceil = 10$ operations whereas our method yields $\lceil \log_2 8 \rceil = 3$ operations.
- List intersection requires analyzing the entire longer adjacency list (*h*) against the much shorter one (*l*). We avoid doing such a large, computationally intensive search every time.

NJIT New Jersey Institute of Technology

What are connected components?



- Connected subgraphs of a graph that is not part of a larger connected subgraph.
- If u is in connected component 1 and v is in connected component 2, there is so possible path $u \rightarrow v$.



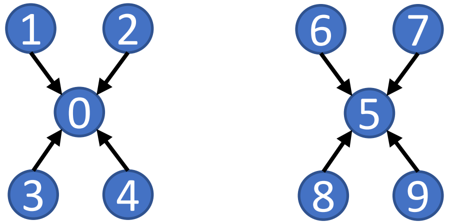
Fast-Spreading Connected Components (1/2)

Algorithm 2: Voltage-Mapping based Fast-Spreading Algorithm

```
FastSpreading(G)
/* G = E,V is the input graph with edge set E and vertex set V.  m = |E| is the total number of edges and n = |V|
   is the total number of vertices.                                                                                */
forall i in 0..n-1 do
 |  L[i] = i
 |  L_n[i] = i
end
/* Initialize the label array L, L_n                                                                               */
while (there is any change in L) do
 |  forall (e = u, v ∈ E) do
 |   |  VO^2(L_n, L, u, v)
 |  end
 |  L = L_n
end
return L
```

$$VO^2(L_n, L, u, v) : \begin{bmatrix} L_n[u] \\ L_n[v] \\ L_n[L[u]] \\ L_n[L[v]] \end{bmatrix} \xleftarrow{ge} ev^2$$

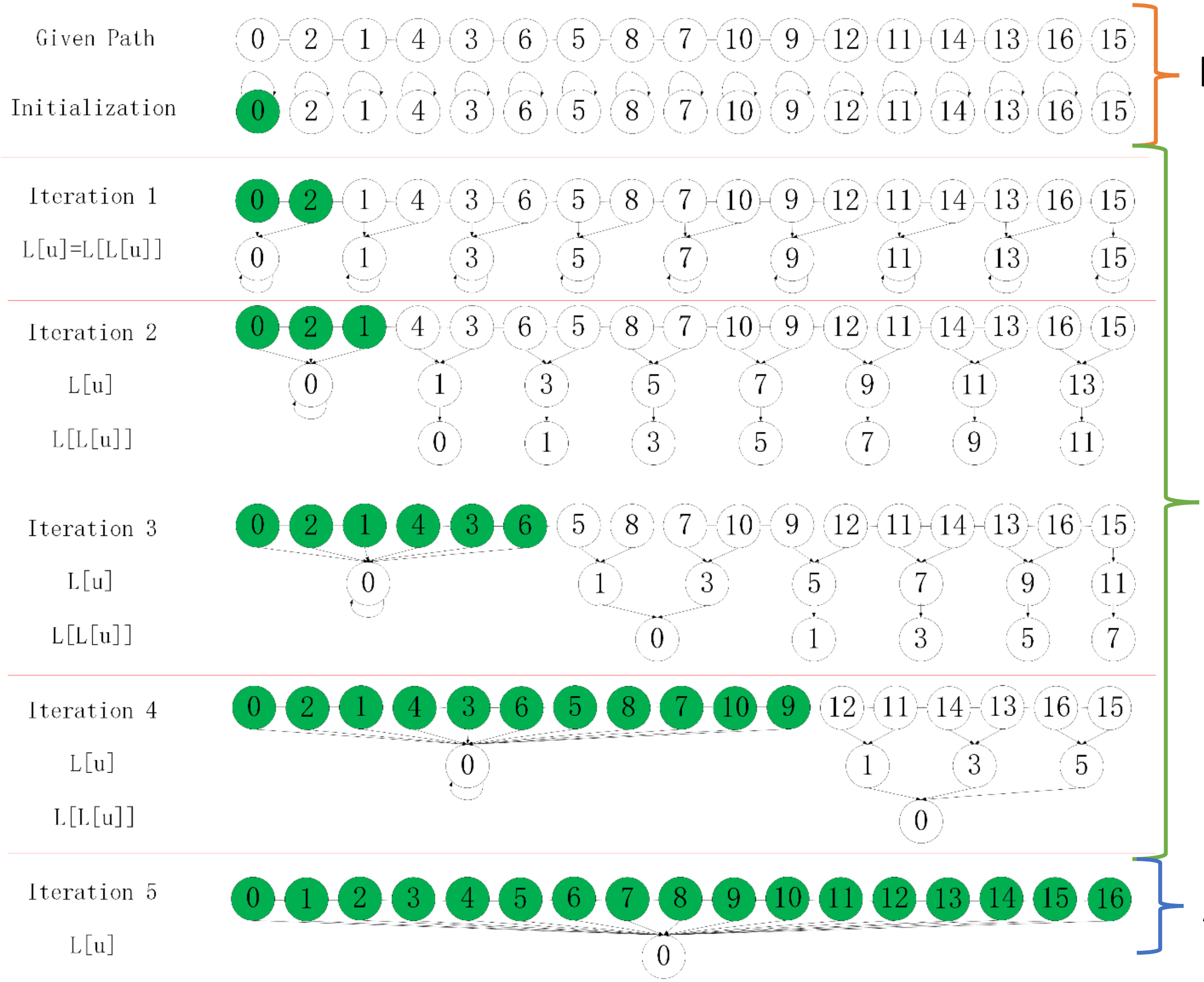
Eventually L will make a graph that looks like this:



$L =$

0	0	0	0	0	5	5	5	5	5

Fast-Spreading Connected Components (2/2)



Initialization, each vertex points to itself

$$VO^2(L_n, L, u, v) : \begin{bmatrix} L_n[u] \\ L_n[v] \\ L_n[L[u]] \\ L_n[L[v]] \end{bmatrix} \xleftarrow{ge} ev^2$$

Propagate the minimum label throughout iterations.

Number of iterations $\log_2 16 + 1 = 5$

All vertices point to smallest.

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
```

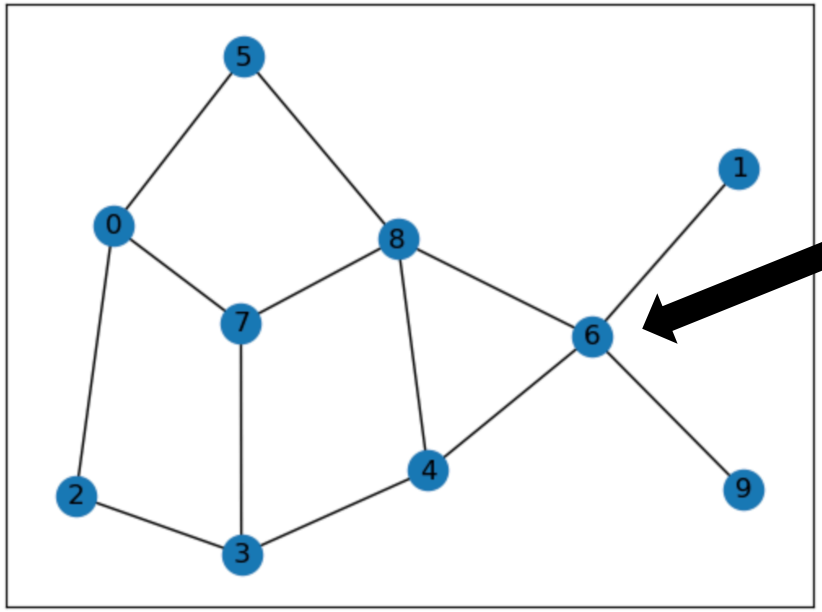
In [4]:

```
# 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
```

Using Arachne with Arkouda (3/3)

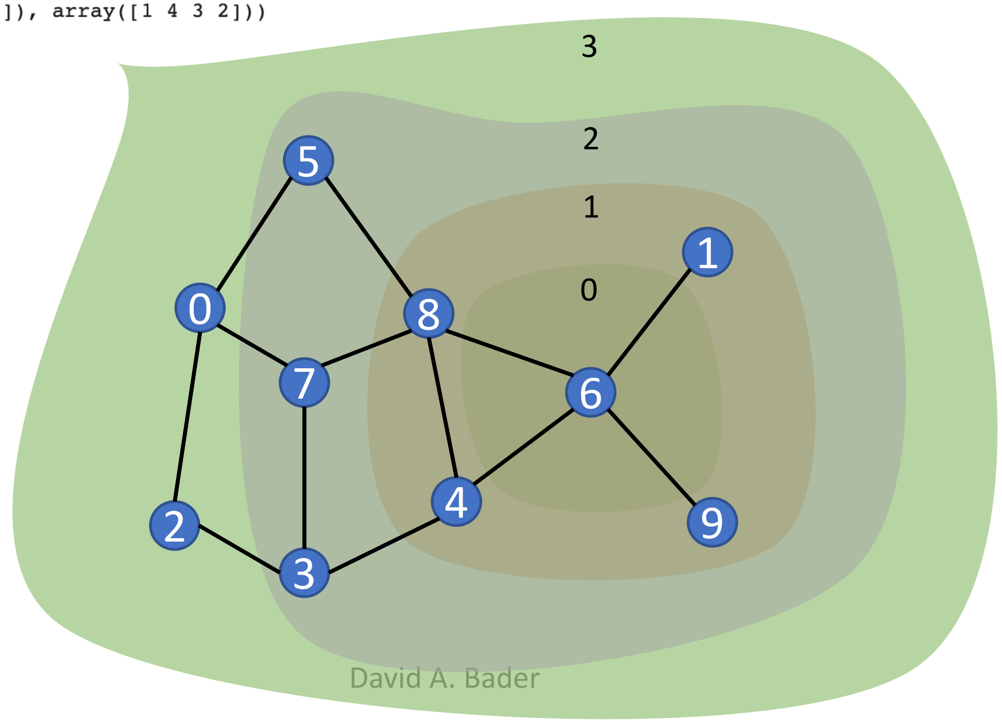
Breadth-First Search

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

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

In [8]:
```python
# 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

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

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

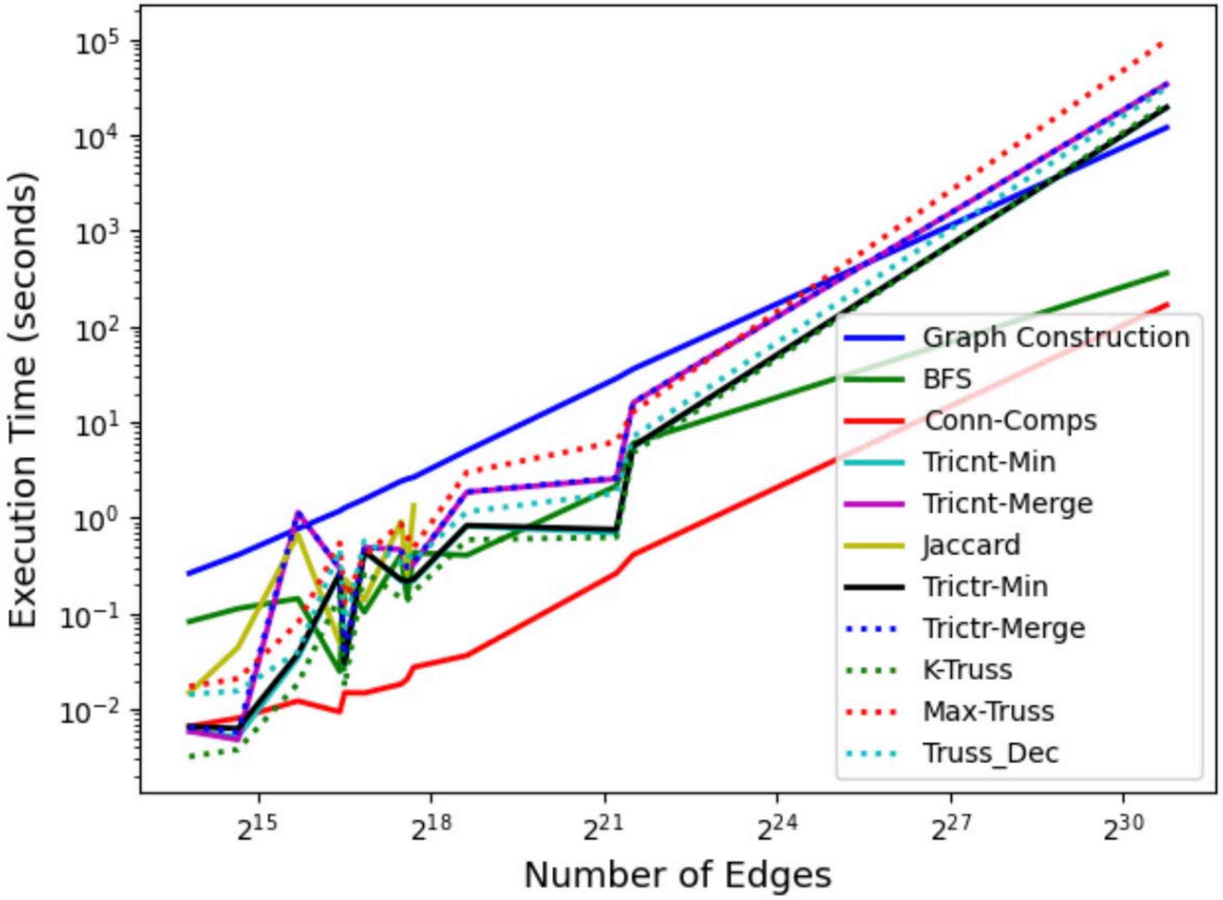
Synthetic

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

NJIT New Jersey Institute of Technology

Arachne Results – Real-World Graphs

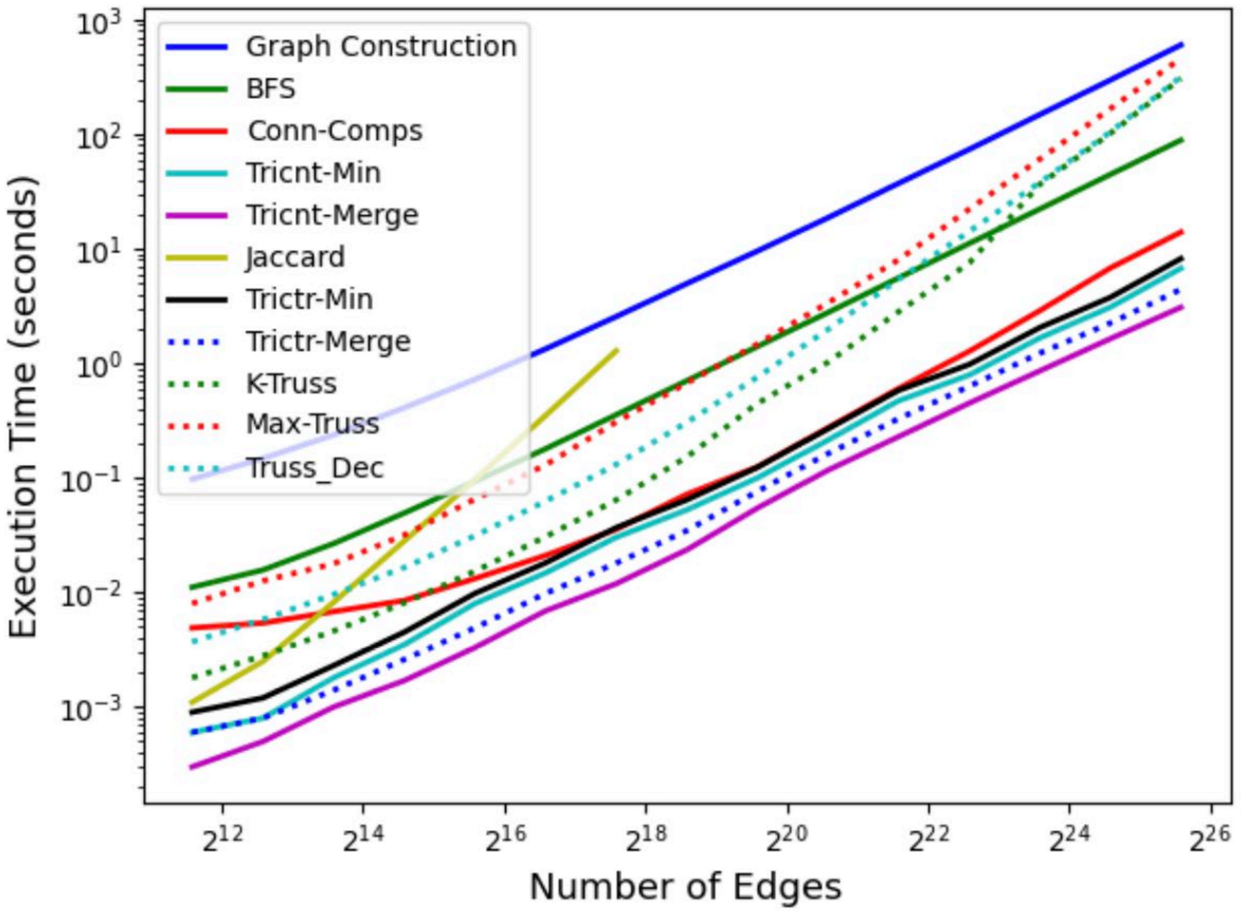
[Alvarado Rodriguez, Du, Patchett, Li, Bader 2022]



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.

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

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

Publications

- **Oliver Alvarado Rodriguez, Zhihui Du, Joseph Patchett, Fuhuan Li, David Bader (2022). Arachne: An Arkouda Package for Large-Scale Graph Analytics. 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.

NJIT
New Jersey Institute of Technology

Acknowledgments

- Dr. Zhihui Du, NJIT
- Oliver Alvarado Rodriguez, NJIT
- Fuhuan Li, NJIT
- Recent Bader Alumni:
  - **Joseph Patchett** (NJIT)
  - **Dr. Eisha Nathan** (Lawrence Livermore National Lab)
  - **Dr. Vipin Sachdeva** (IBM)
  - **Dr. Anita Zakrzewska** (Trovares)
  - **Dr. Lluis Miquel Munguia** (Google)
  - **Prof. Kamesh Madduri** (Penn State)
  - **Dr. David Ediger** (GTRI)
  - **Prof. James Fairbanks** (University of Florida)
  - **Dr. Seunghwa Kang** (NVIDIA)