Graph Analytics in Arkouda

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, SIAM Fellow, AAAS Fellow
- IEEE Computer Society Sidney Fernbach Award
- Recent Service:
    - White House's National Strategic Computing Initiative (NSCI) panel
    - Computing Research Association Board
    - 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 $185M of research awards
- 300+ publications, ≥ 11,000 citations, h-index ≥ 61
- 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

Launched in **July 2019**, with inaugural director
David A. Bader
(~35 faculty in current centers)

NJIT Data Science Seminar Series
Wednesday’s 4pm ET
https://njit-institute-for-data-science.eventbrite.com/

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
- Planned launch: 2021

NJIT New Jersey Institute of Technology

Upcoming NJIT Data Science Seminars

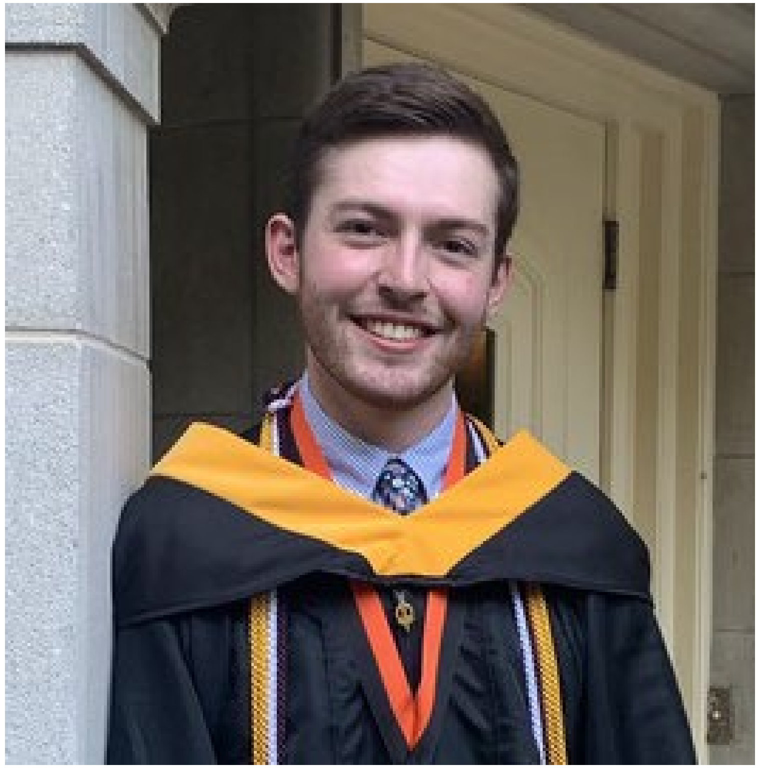
https://datascience.njit.edu/events

- October 6: Gordon Bell
- October 20: Jack Dongarra
- October 27: Larry Smarr
- November 3: Laura Haas

YouTube

https://www.youtube.com/c/NJITInstituteforDataScience/

NJIT
New Jersey Institute of Technology



Joint work with

- Dr. Zhihui Du (NJIT)
- Oliver Alvarado Rodriguez (PhD student)
- Joseph Patchett (MS student)

High Performance Algorithms for Interactive Data Science at Scale

(PI: Bader) 3/2021 – 2/2022, NSF CCF-2109988

A real-world challenge in data science is to develop interactive methods for quickly analyzing new and novel data sets that are potentially of massive scale. This award will design and implement fundamental algorithms for high performance computing solutions that enable the interactive large-scale data analysis of massive data sets.

This project focuses on these three important data structures for data analytics:
1) suffix array construction,
2) ‘treap’ construction, and
3) distributed memory join algorithms,

useful for analyzing large scale strings, implementing random search in large string data sets, and generating new relations, respectively.

To evaluate and show the effectiveness of the proposed algorithms, these algorithms will be implemented in and contribute to an open source NumPy-like software framework that aims to provide productive data discovery tools on massive, dozens-of-terabytes data sets by bringing together the productivity of Python with world-class high performance computing.

Institute for Data Science Aims to Democratize Supercomputing With NSF Grant

Written by: Evan Koblentz

Published: Wednesday, March 17, 2021

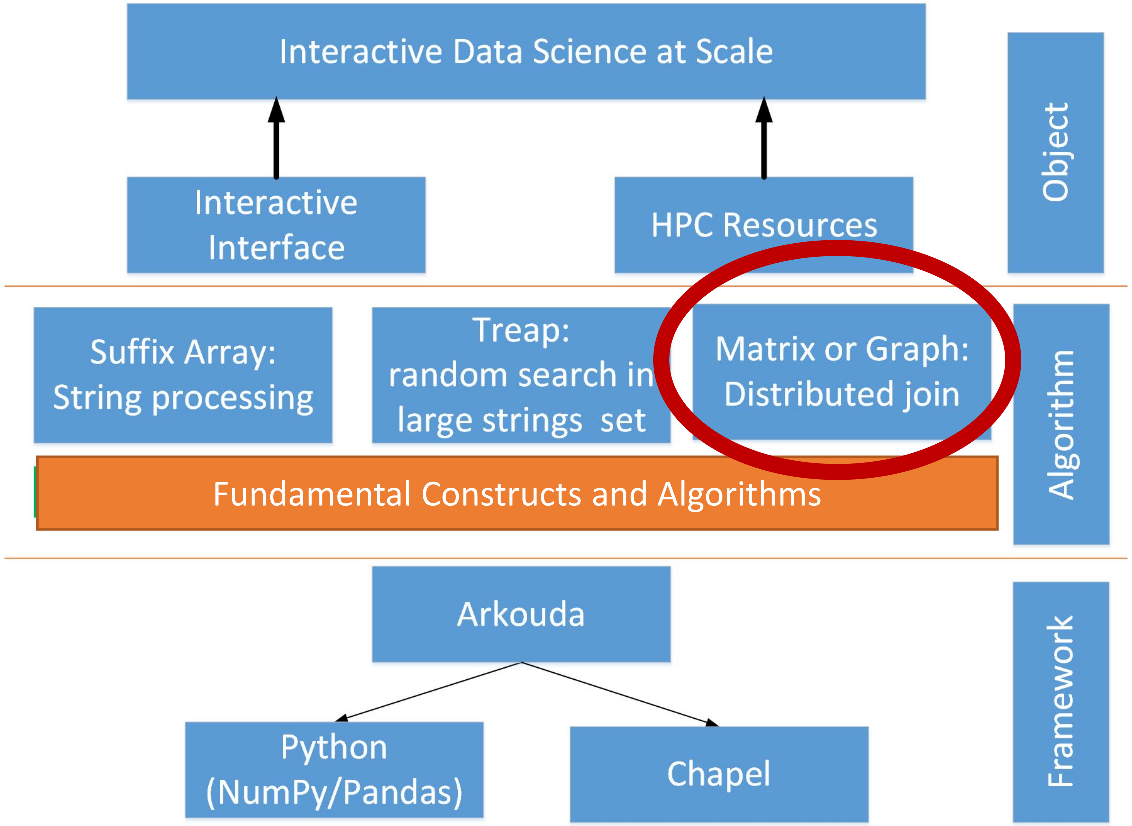
New algorithms from at NJIT can make supercomputer power available to almost anyone

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

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

Interactive Data Science at Scale

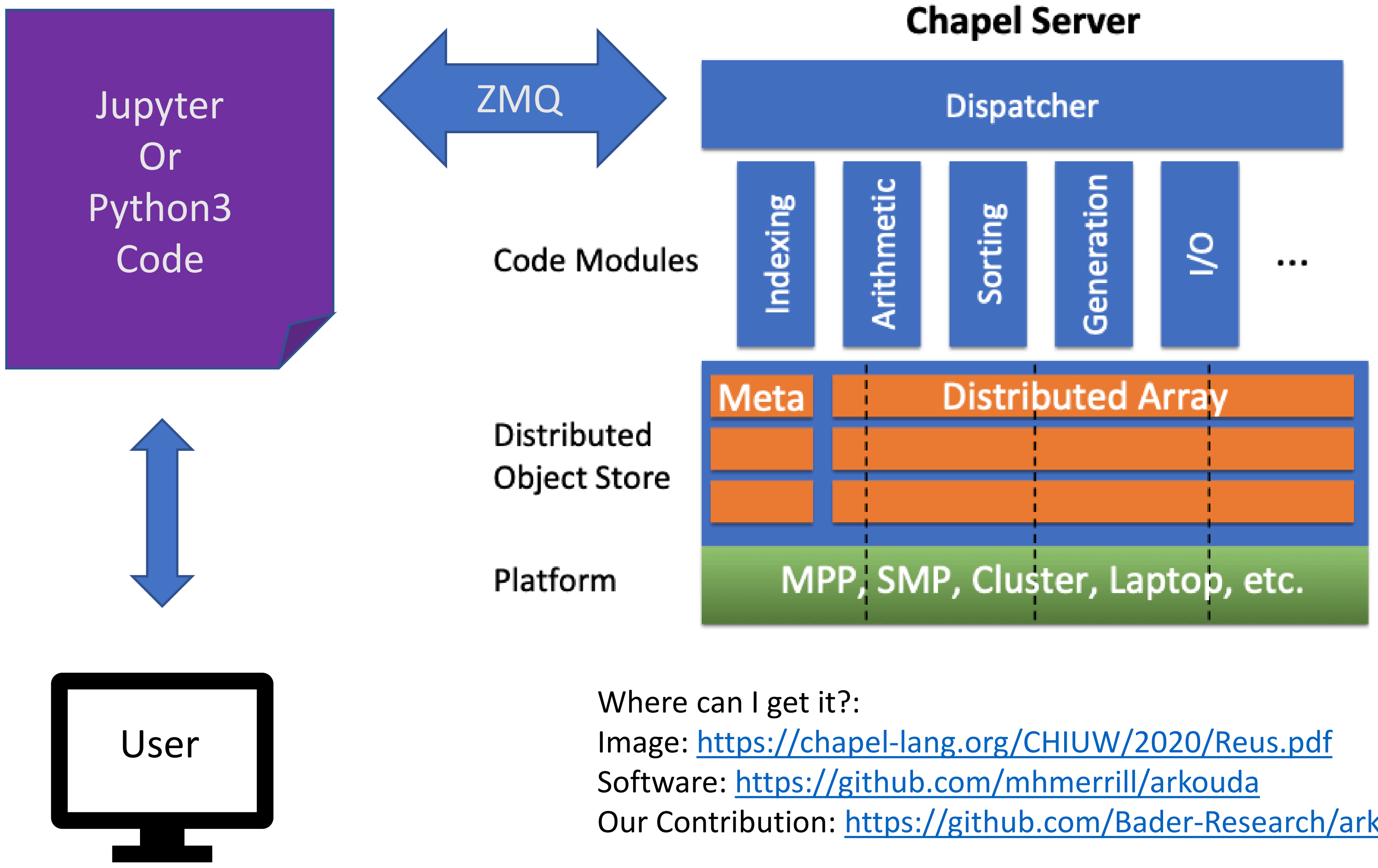
- Objective:
  - One-stop solution for non-HPC experts with massive data sets.
- Developmental focus:
  - Data structures and algorithms for graph and other problems.
- Framework:
  - Arkouda



New Jersey Institute of Technology

Productivity + Performance

Typical Environment Set-Up



Python3 Implementation:

- Pdarray class
- Rely on Python to reduce complexity
- Integrate with and use NumPy

Server Implementation:

- High-level language with C-comparable performance
- Great parallelism handling
- Great distributed array support
- Portable code: laptop --> HPC

Where can I get it?:
Image: https://chapel-lang.org/CHIUW/2020/Reus.pdf
Software: https://github.com/mhmerrill/arkouda
Our Contribution: https://github.com/Bader-Research/arkouda/tree/streaming

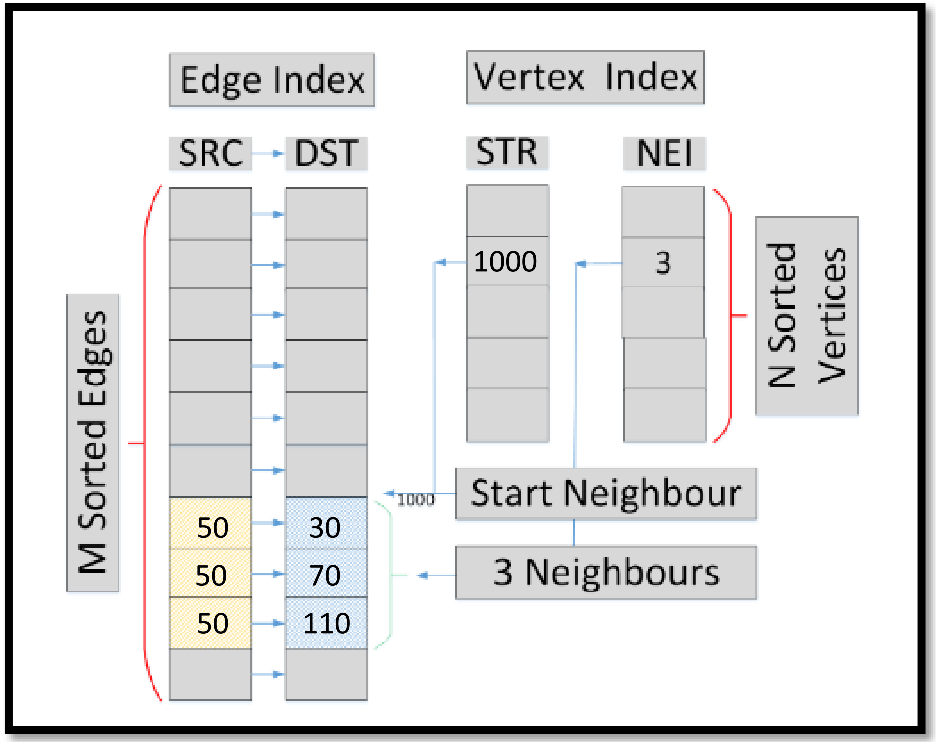
NJIT New Jersey Institute of Technology

Large-Scale Graph Analytics in Arkouda

- Graph Data Structure (Double-Index: DI)
  - Edge-oriented partitioning scheme
  - [Du, Alvarado Rodriguez, Bader 2021]
- Breadth-First Search
  - A high-level and low-level distributed and parallel implementation of breadth-first search (BFS) with reverse Cuthill-McKee (RCM) relabeling heuristic if needed.
  - [Alvarado Rodriguez, Du, Bader 2021]
- Suffix Array
  - Highly-productive: Python users can process massive collections of strings in almost the same way as small strings
  - [Du, Alvarado Rodriguez, Bader 2021]
- K-Truss
  - Design an optimized multi-locale (distributed) parallel k-Truss algorithm that can significantly improve the performance
  - [Patchett, Du, Bader, 2021]

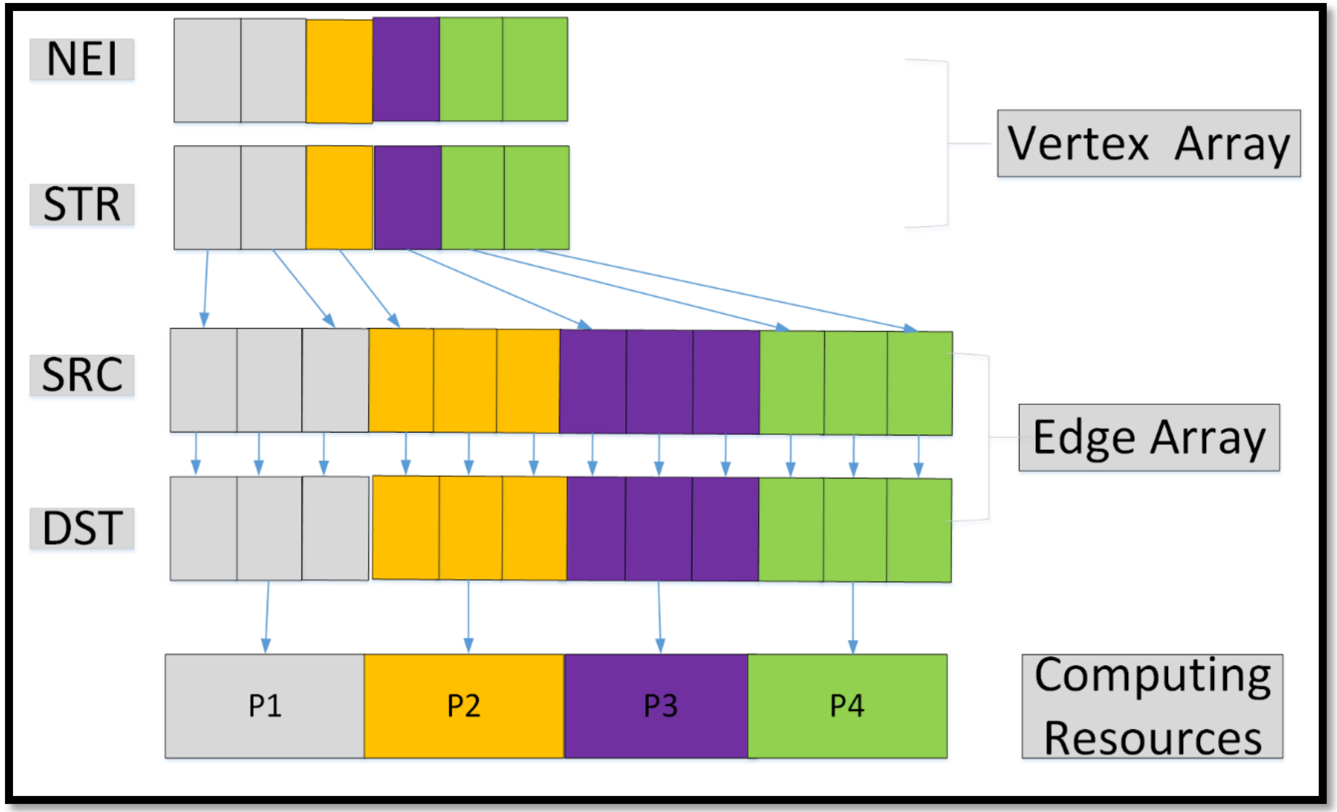
Double-Index Graph Data Structure

- Advantage
  - O(1) time complexity
    - Locate specific vertex from given edge ID
    - Locate adjacency list from given vertex ID
- Comparison with CSR
  - Similarity
    - Value array <-> SRC/DST, array size is NNZ
    - Column index <-> STR, array size is |V|
    - Row index <-> NEI, array size is |V|+1 and |V|
  - Difference
    - We can search from edge ID to vertex ID, CSR cannot.
    - We need a bit more memory (another NNZ array) than CSR



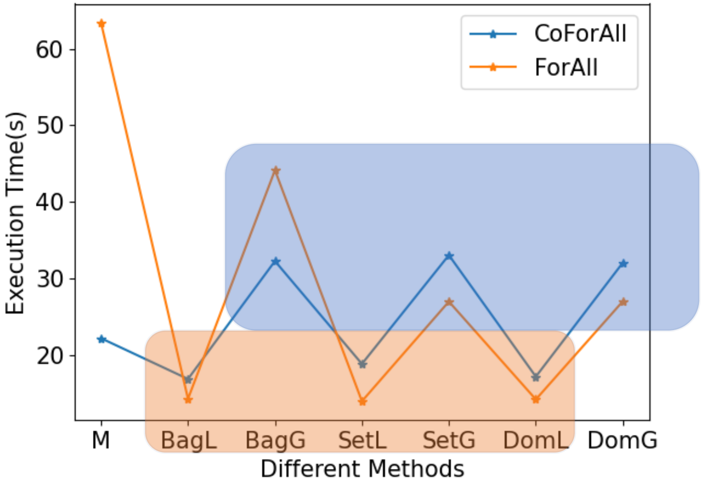
New Jersey Institute of Technology

Support Edge-Oriented Graph Partition

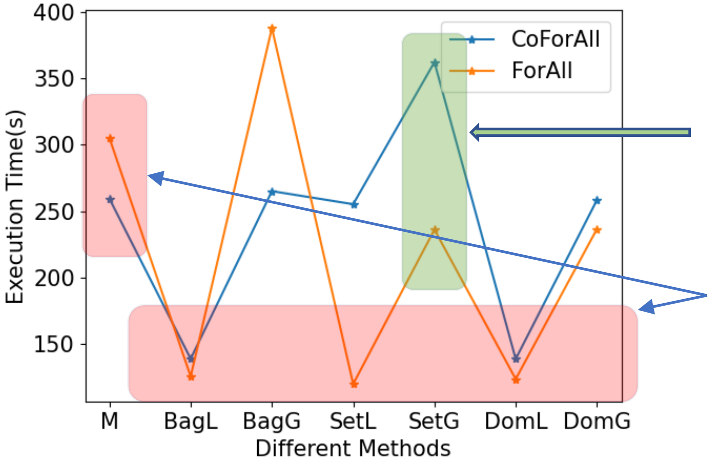


- Imbalance in vertex's degree
  - Power law degree distribution of real-world graphs
    - One vertex can have very different number edges
- Edge oriented partition
  - Edge array is much larger than vertex array
    - Align vertices to corresponding edge
    - load balancing

Breadth-First Search Algorithms in Arkouda



Delaunay 17



Delaunay 20

- High level data structure
  - Distributed bag (Bag), set (Set), and domain (Dom)
  - L is the distributed parallel computing version. G is the shared memory computing version.
  - M is the low-level version
- Parallel construct
  - forall/coforall

redundant calculations without idle threads

no redundant calculation with idle threads

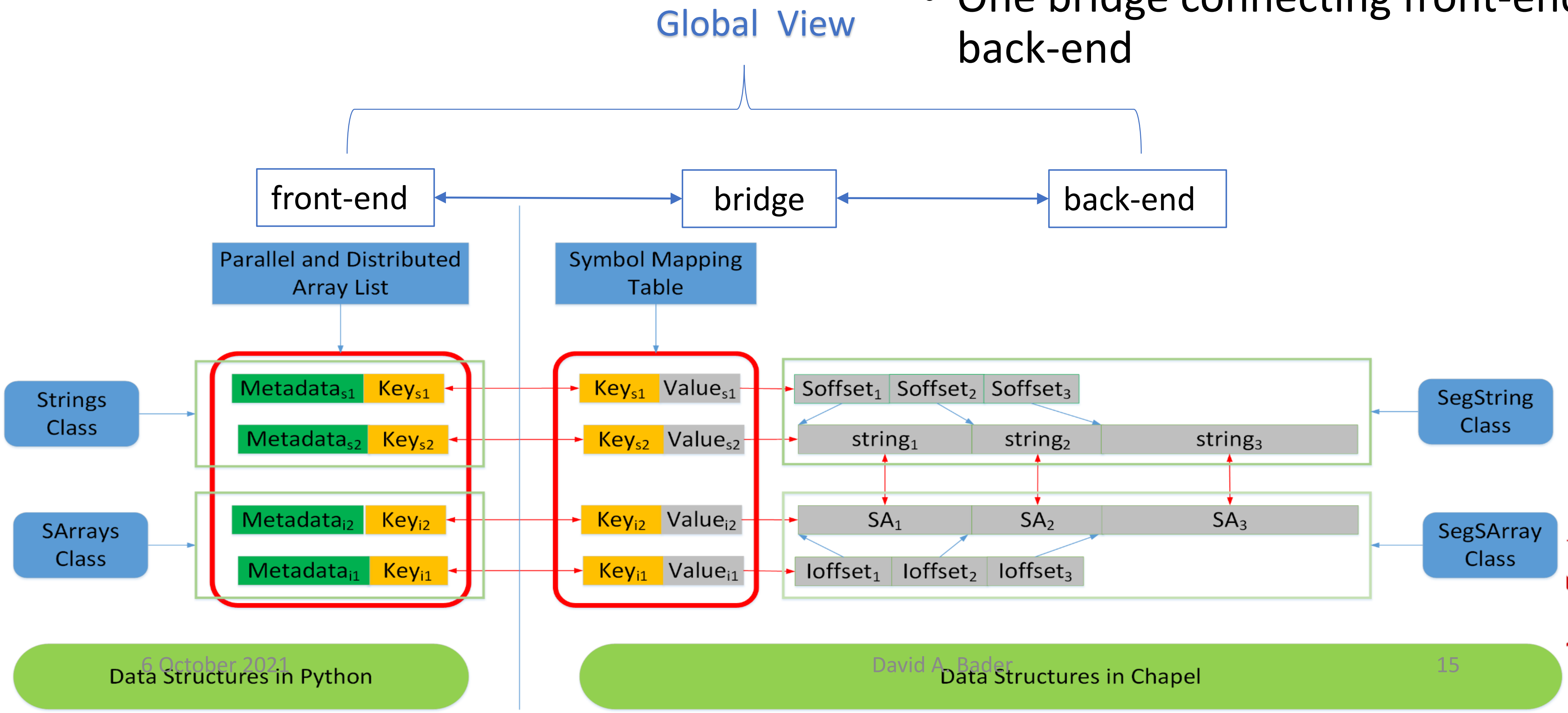
Different parallel constructs can affect performance

High-level algorithm can compete with low-level algorithm under the same algorithm framework

New Jersey Institute of Technology

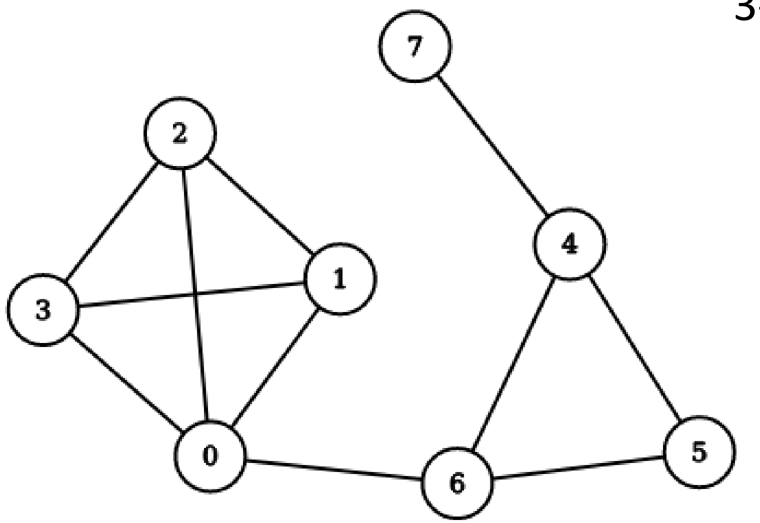
Suffix arrays for a group of strings

- Example
  - Input strings SG={“cba”,”5986” }
  - Output suffix arrays SA={[3,2,1],[1,4,3,2]}
  - Value/offset arrays at back-end
- GLB (Global-Local-Bridge) design
  - One global view
  - Two local implementations
  - One bridge connecting front-end and back-end

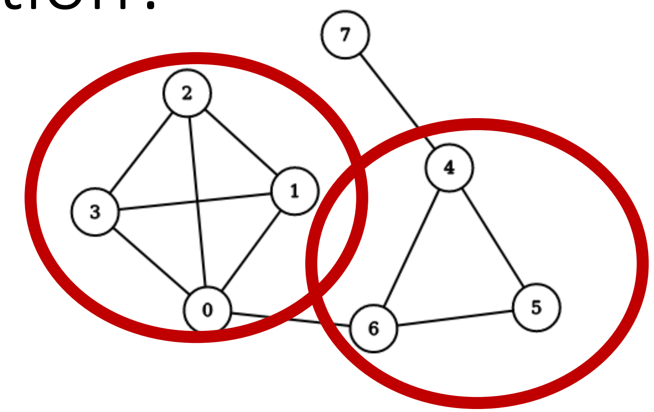


New Jersey Institute of Technology

Can you find the K-Truss Decomposition?

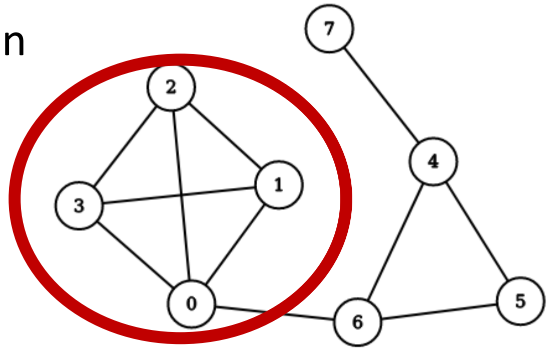


3-Truss Decomposition



How does our k-Truss work though?

4-Truss Decomposition



New Jersey Institute of Technology

High Level k-Truss Algorithm

```
G_prime = preprocess(G)
/* G is the graph defined by our double
   index data structure                    */
while A change has been made do
    TriCount = T(G_prime)
    //Compute the Initial triangle counts
    Initialize current frontier SetCurF
    Initialize next frontier SetNextF
    if Edge m has TriCount[m] < k − 2 then
        Assign m to SetCurF
        Remove m from G_prime
    end
    coforall (loc in Locales ) do
        while SetCurF is not empty do
            forall Edges < u, v > in the SetCurF do
                forall Adjacent edges do
                    if A triangle exists then
                        Decrement TriCount for the
                         edge
                        if Edge has TriCount < k-2
                         then
                            Add the edge to SetNextF
                            Remove the edge from
                             G_prime
                        end
                    end
                end
            end
            SetCurF <=> SetNextF
            clear SetNextF
        end
    end
end
return G_prime
```

Some edges aren't part of any triangles and can be removed

The frontier maintains edges to be removed

Use DI graph structure to quickly assess adjacency

New Jersey Institute of Technology

Acknowledgement

We appreciate the help from the Chapel and Arkouda teams including: Brad Chamberlain, Elliot Joseph Ronaghan, Engin Kayraklioglu, David Longnecker, Michael Merrill, and Bill Reus.

This research was funded in part by NSF grant number CCF-2109988.

New Jersey Institute of Technology