# PageRank Pipeline Benchmark: Proposal for a Holistic System Benchmark for Big-Data Platforms

**Patrick Dreher[1,4], Chansup Byun[2], Chris Hill[3], Vijay Gadepally,[1,2]**
**Bradley C. Kuszmaul[1], Jeremy Kepner[1,2]**

[1]MIT Computer Science & AI Laboratory; [2]MIT Lincoln Laboratory Supercomputing Center; [3]MIT Department of Earth, Atmospheric and Planetary Sciences; [4]Department of Computer Science, North Carolina State University

**Graph Algorithms Building Blocks Workshop (GABB 2016)**
**30th IEEE International Parallel & Distributed Processing Symposium**
**Chicago, IL May 2016**

Massachusetts Institute of Technology

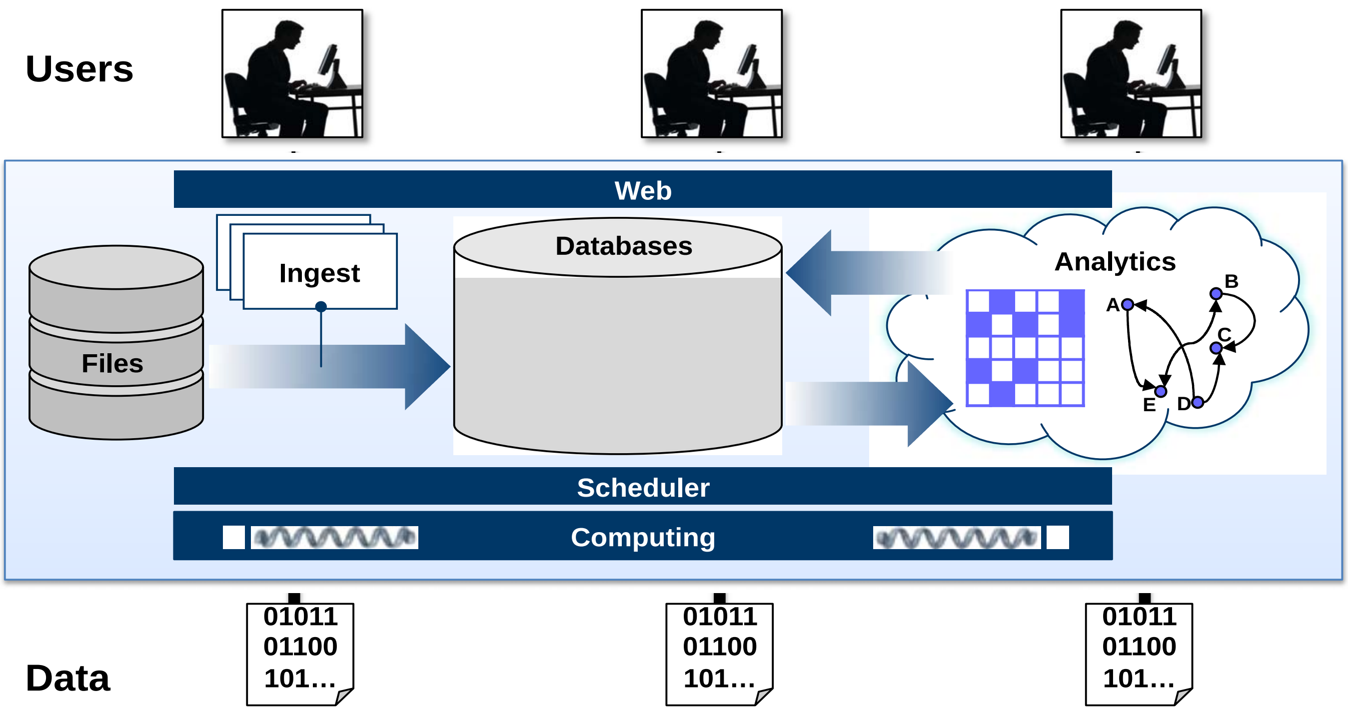
# Outline

- Growth of Big Data and the Value of Information
- Big Data Attributes
- Benchmarking Big Data Systems
- Benchmark Shortcomings and Ambiguities
- Development of a Simple Big Data Benchmark
- Results
- Summary – Next Steps

# Growth of Big Data and the Value of Information

- Processing/analysis of data is an essential aspect of many domain/subject matter areas
- Data itself is witnessing large increases in
  - Volume – amount of data
  - Velocity - rate at which data is being collected
  - Variety/types – characteristics and properties of the data
  - Variability – complex time dependent changes among volume, variety and variability
- Recognized that valuable information is contained in the data
- To access that information need to develop
  - hardware architectures
  - software environments
- Must validate these big data systems with reliable benchmarks

# Common Architecture for Connecting Diverse Data and Users






*PageRank Pipeline Benchmark: Proposal for a Holistic System Benchmark for Big-Data Platforms*, Dreher et al, IPDPS GABB 2016

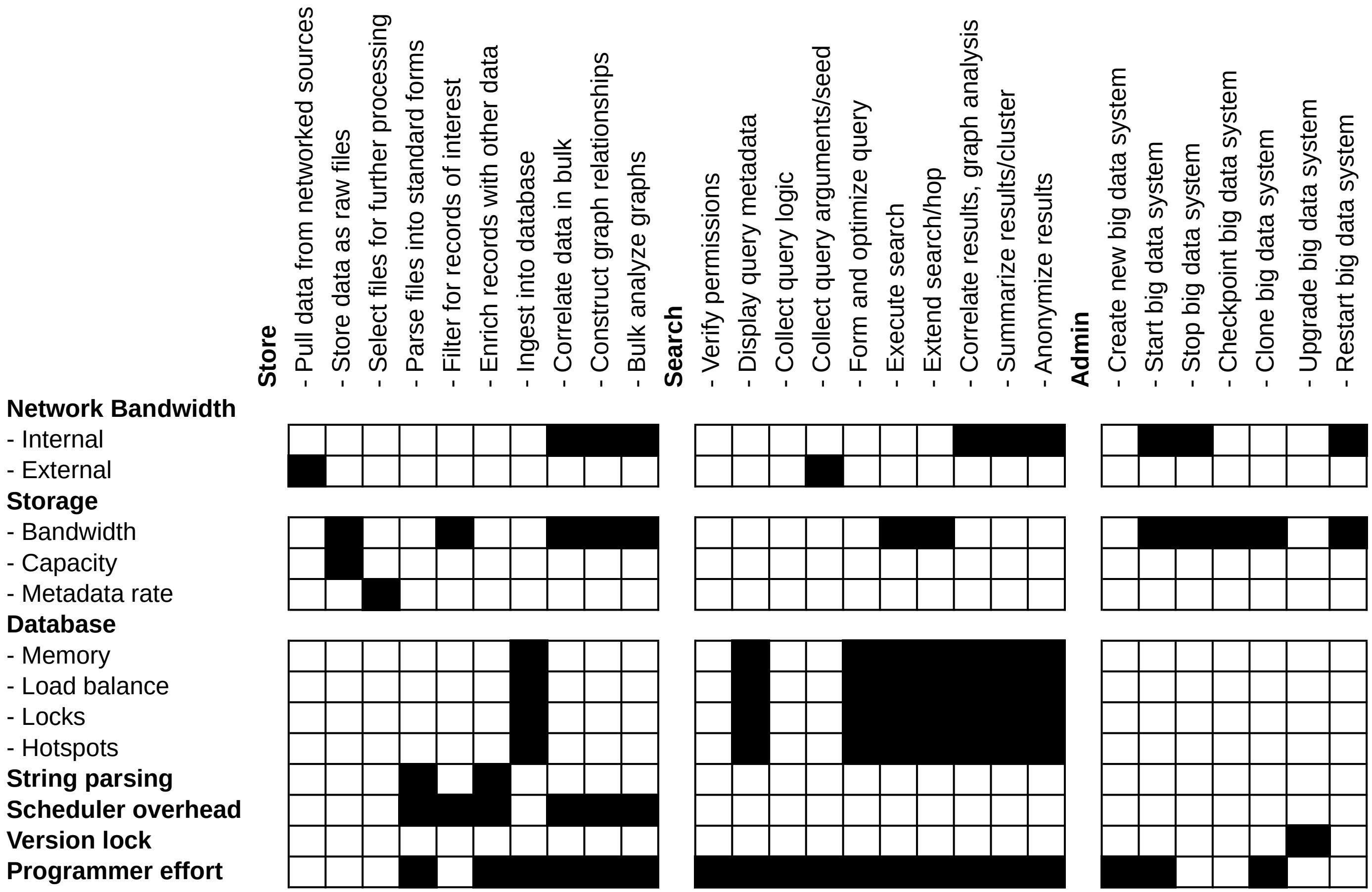
# High Performance Data Analysis Attributes

**Store**
- Pull data from networked sources
- Store data as raw files
- Select files for further processing
- Parse files into standard forms
- Filter for records of interest
- Enrich records with other data
- Ingest into database
- Correlate data in bulk
- Construct graph relationships
- Bulk analyze graphs

**Search**
- Verify permissions
- Display query metadata
- Collect query logic
- Collect query arguments/seed
- Form and optimize query
- Execute search
- Extend search/hop
- Correlate results, graph analysis
- Summarize results/cluster
- Anonymize results

**Admin**
- Create, start, stop, checkpoint, clone, upgrade, restart, …

# Workload Analysis Bottlenecks




- **Large number of existing Big Data benchmarks**
- **Shortcoming is that most are easily tuned and therefore have a weak correlation with application performance**



*PageRank Pipeline Benchmark: Proposal for a Holistic System Benchmark for Big-Data Platforms*, Dreher et al, IPDPS GABB 2016

# Goal: Develop Benchmark Performance That Correlates with Application Performance

- **HPC community benchmarks have**
  - **Long tradition of developing various methodologies for creating rigorous benchmarks for hardware architectures and software environments**
  - **Emphasize performance and scalability**

- **Develop similar rigorous methodologies for creating data intensive benchmark(s) that**
  - **Test both the hardware architecture and software systems**
  - **Amenable to implementation in diverse environments**
  - **Reflect realistic workflows**
    - **Incorporate kernels emphasizing reads, writes, sorts and shuffles**
    - **Fully measure the substantial extract-transform-load costs of data movement prior to focusing on higher-order benchmark kernels**

# Select a Benchmark Appropriate to Measure Big Data Application Performance

- Build a big data benchmark from among a choice of four types of benchmark categories
  - Goal-oriented (Graph500 Sort [a])
  - Algorithm-oriented (NAS [b])
  - Code-oriented (Top500 [c], HiBench [d])
  - Standards-oriented (HPC Challenge [e])
- Selected algorithm-oriented benchmark category
  - Allows maximum flexibility to test total system implementation
  - Allows re-implementation in diverse environments
  - Can benchmark both hardware and software

[a] http://www.graph500.org/

[b] https://www.nas.nasa.gov/Software/NPB/

[c] http://www.top500.org/project/

[d] https://www.ibm.com/support/knowledgecenter/SSGSMK_7.1.1/mapreduce_integration/map_reduce_hibench.dita

[e] http://icl.cs.utk.edu/hpcc/

# PageRank Pipeline Algorithm

- **PageRank selected because of algorithm's inherent simplicity and generality**
  - **Builds on existing prior scalable benchmarks (Graph500, Sort, and PageRank)**
  - **Well defined mathematically and can be implemented in any programming environment**
  - **Provides rigorous definition of both the input and the output for each kernel**
  - **Emulates data operations not solely governed by the CPU speed in the hardware platform**
  - **Quantitatively compare a wide range of present day and future systems because it is scalable in both problem size and hardware**

- **Constructs a data pipeline flow that**
  - **Creates a holistic benchmark with multiple integrated kernels**
  - **Implements ordered set of kernels with reads, writes, sorts and shuffles with process characteristics and similarities to big data applications**
  - **Kernels can be run together or independently**
  - **Reflects characteristics many data analytics workloads**
  - **Can be used to build a whole-system benchmark focused toward measuring performance of emerging Big-Data architectures**



*PageRank Pipeline Benchmark: Proposal for a Holistic System Benchmark for Big-Data Platforms*, Dreher et al, IPDPS GABB 2016

# PageRank Pipeline Benchmark

- **Construct a pipeline sequence of four benchmark kernels based on the PageRank algorithm that can mimic the full workload required to perform PageRank on a random graph**

  - **Kernel 0**

    **generate graph edges (Graph 500* generator) and writes output to storage**

  - **Kernel 1**

    **Read files from Kernel 0, sort edges by start vertex, write to non-volatile storage**

  - **Kernel 2**

    **Read files from Kernel 1, construct adjacency matrix**
    **Compute in-degree and eliminate high and low degree nodes**
    **Normalize each row by total number of edges in row**
    **Weight the sparse matrix values**

  - **Kernel 3**

    **From output of Kernel 2 perform 20 iterations of PageRank on normalized adjacency matrix (sparse matrix vector multiply)**

* D. Bader, K. Madduri, J. Gilbert, V. Shah, J.y Kepner, T. Meuse, and A. Krishnamurthy, “Designing Scalable Synthetic Compact Applications for Benchmarking High Productivity Computing Systems,” CT Watch, Vol 2, Number 4A, November, 2006.

# PageRank Pipeline Benchmark
# Serial Code Reference Implementations#

| Language | Source Lines of Code |
|---|---|
| C++ | 494 |
| Python | 162 |
| Python w/Pandas | 162 |
| MATLAB | 102 |
| Octave | 102 |
| Julia | 162 |

- ~10 lines of math
- Easy to implement
- References (listed below) for implementation in many popular languages *

# Intel Xeon E5-2650 (2 GHz) (16 cores) with 64 Gbytes of memory and InfiniBand and 10 GigE interconnects

* The source code listing for the PageRank Pipeline Benchmark in each of the languages (C++, Julia, MATLAB, Python and Octave) is located here
https://github.com/vijaygadepally/PageRankBenchmark/tree/master/code

* There is a README.txt with information how to run the benchmark that is located here
https://github.com/vijaygadepally/PageRankBenchmark/blob/master/README.txt

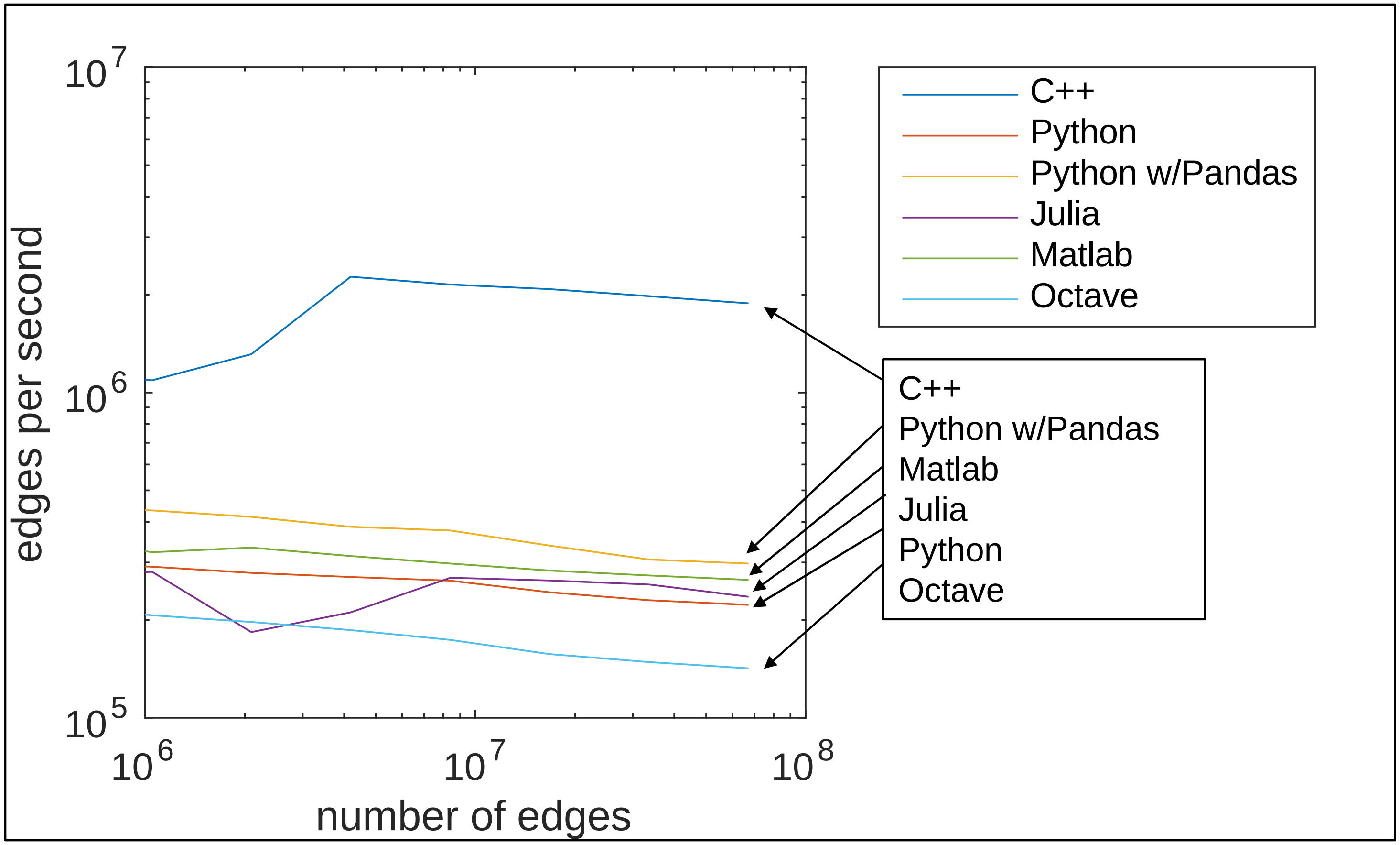
# Measured Problem Size

- There are 2 inputs to the PageRank Pipeline Benchmark Algorithm
  - Scale factor *S* that determines maximum number of vertices
  - Edges per vertex factor *k*

- Maximum number of vertices $N = 2^S$
- Maximum number of edges = kN
- The scale and vertex factors determine the overall size of the graph
- The speed of the sort ordering varies depending on the matrix size
- Scale sizes chosen sufficiently large to limit any L3 cache advantage for in-memory computations

| Scale | Max Vertices | Max Edges | ~Memory |
|---|---|---|---|
| 16 | 65K | 1M | 25MB |
| 17 | 131K | 2M | 50MB |
| 18 | 262K | 4M | 100MB |
| 19 | 524K | 8M | 201MB |
| 20 | 1M | 16M | 402MB |
| 21 | 2M | 33M | 805MB |
| 22 | 4M | 67M | 1.6GB |

# Kernel 0: Generate Graph

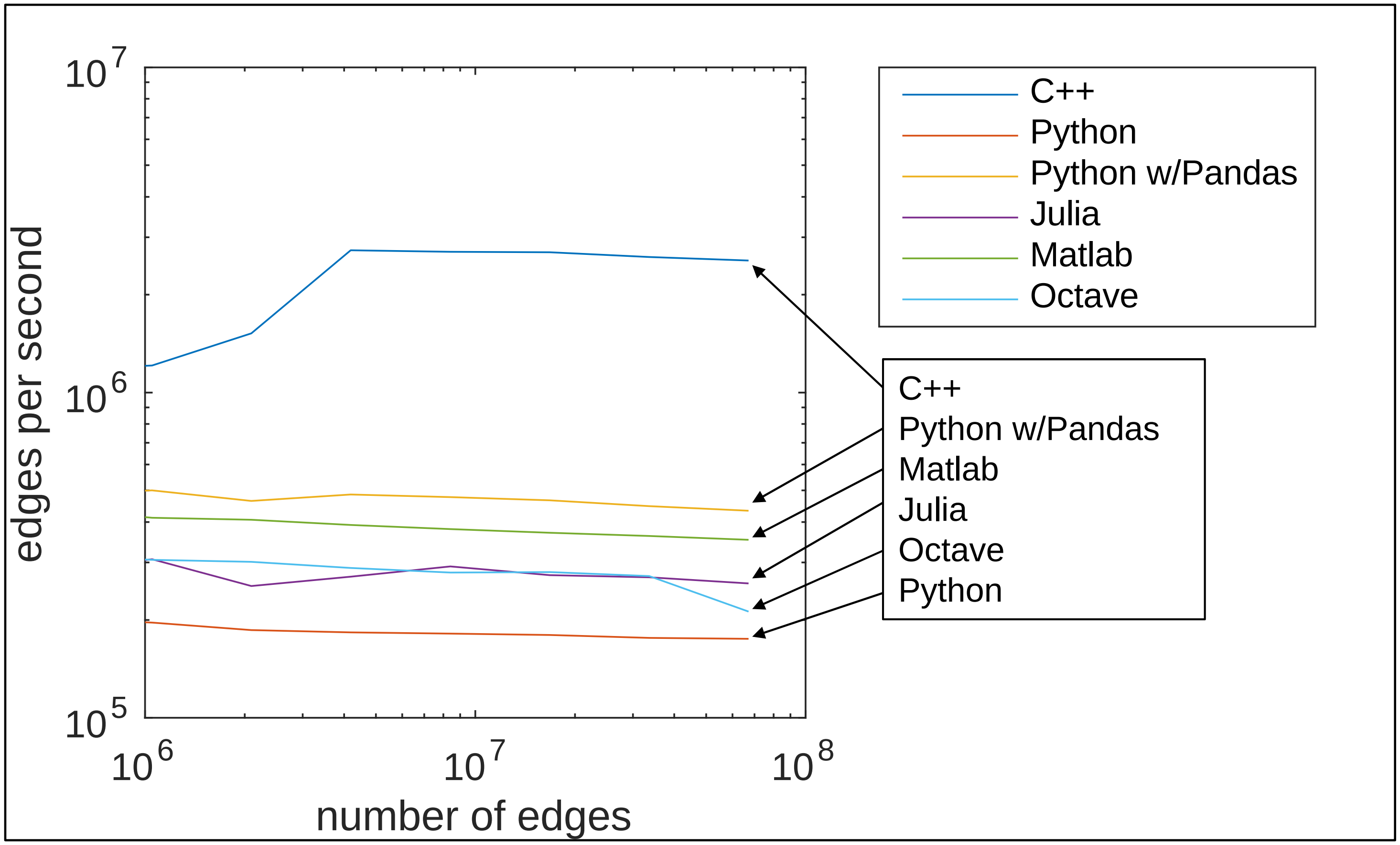



- **Approximately power-law graph**
- **Essentially utilizes algorithm of Graph500 graph generator**
- **I/O Intensive**
- **Untimed**



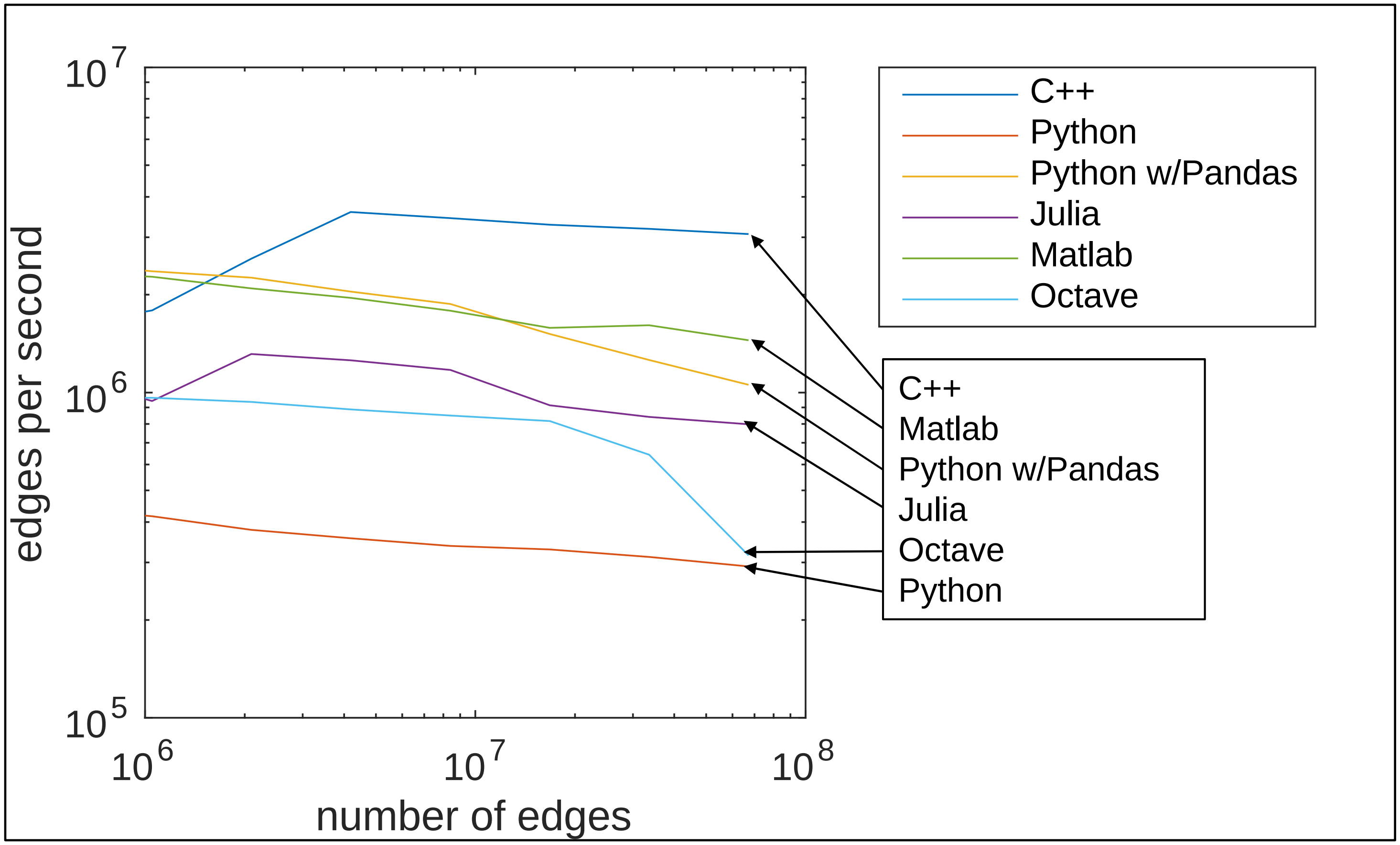
*PageRank Pipeline Benchmark: Proposal for a Holistic System Benchmark for Big-Data Platforms*, Dreher et al, IPDPS GABB 2016

# Kernel 1: Sort Edges




- **I/O intensive**
- **Network intensive**
- **Storage cache may inevitably impact Kernel 1 results**

# Kernel 2: Filter Vertices

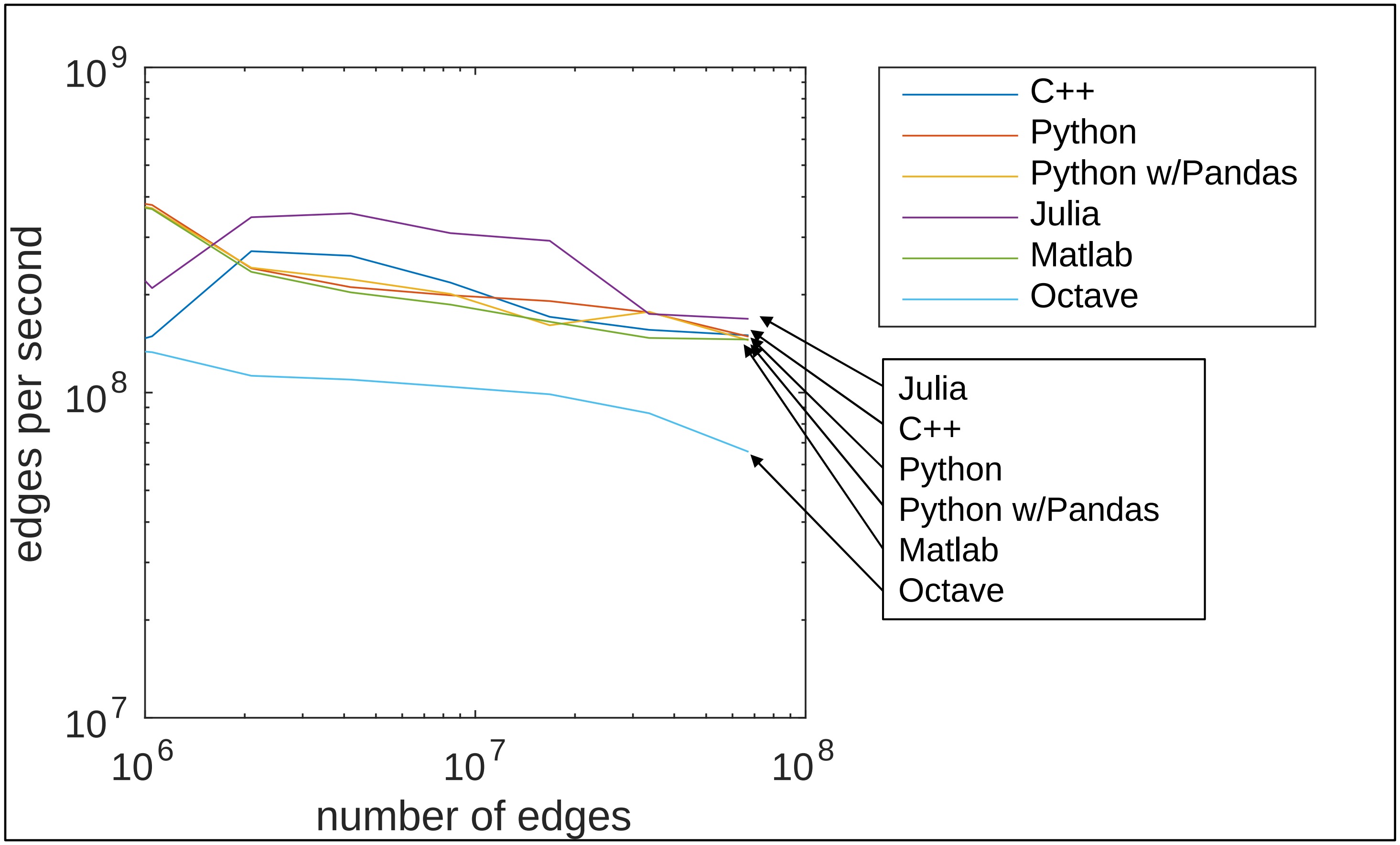



- **I/O intensive**
- **Memory intensive**



*PageRank Pipeline Benchmark: Proposal for a Holistic System Benchmark for Big-Data Platforms*, Dreher et al, IPDPS GABB 2016

# Kernel 3: PageRank




- **Memory intensive**
- **Compute intensive**



*PageRank Pipeline Benchmark: Proposal for a Holistic System Benchmark for Big-Data Platforms*, Dreher et al, IPDPS GABB 2016

# Summary and Next Steps

- **PageRank is useful for benchmarking big data workloads in a variety of hardware architectures and software environments**
- **Allows benchmarks to be measured with variations in platform configurations that include**
  - **Use of local disks versus remote storage**
  - **Various network interconnects among servers**
  - **Different cache sizes in the server**
- **For each type of platform configuration, various sizes of adjacency matrices can be constructed and sorting speeds measured for each type of hardware and software configuration using the PageRank algorithm**
- **Next Steps**
  - **Develop full math specification**
  - **Serial and parallel reference implementations**

# Questions *

* Corresponding author dreher@mit.edu