V3-24: Scalable Computational Analytics

Mission-Critical Computing
NSF CENTER FOR SPACE, HIGH-PERFORMANCE, AND RESILIENT COMPUTING (SHREC)

SHREC Annual Workshop (SAW23-24)

VIRGINIA TECH

January 17-18, 2024

Faculty

Wu Feng

Students & Post-Students

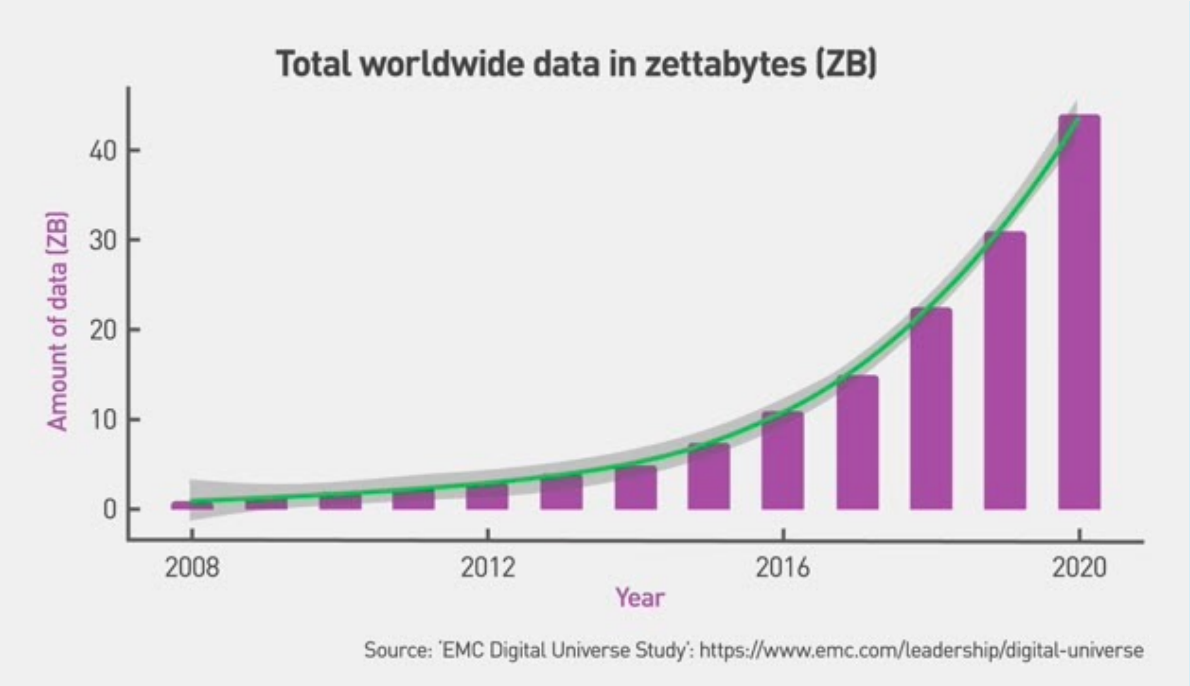
Ritvik Prabhu, Frank Wanye

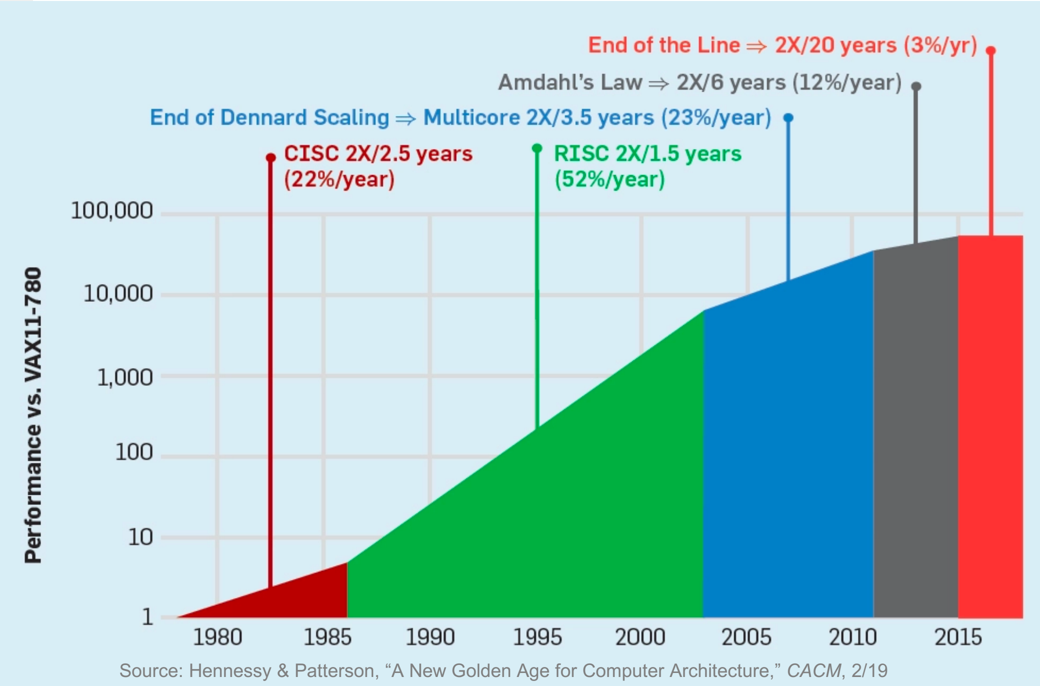
Number of requested memberships ≥ 2

Motivation

V3

- Exponential growth in ***data*** is ***far surpassing*** the growth in ***computing***
  - Need more ***scalable*** approaches (that are efficient & interactive) to process big data

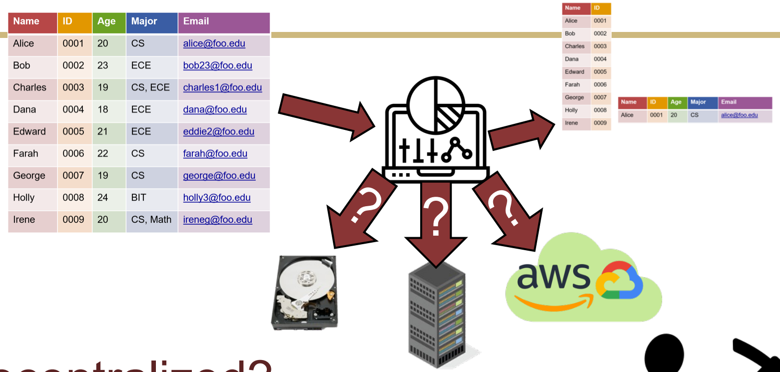




Doubling every 2 years

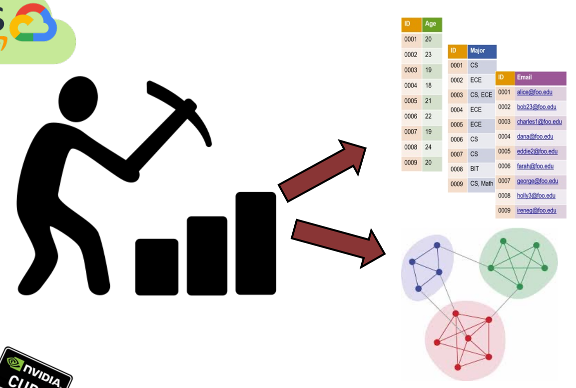
Doubling every 20 years

Motivation



Name	ID	Age	Major	Email
Alice	0001	20	CS	alice@foo.edu
Bob	0002	23	ECE	bob23@foo.edu
Charles	0003	19	CS, ECE	charles1@foo.edu
Dana	0004	18	ECE	dana@foo.edu
Edward	0005	21	ECE	eddie2@foo.edu
Farah	0006	22	CS	farah@foo.edu
George	0007	19	CS	george@foo.edu
Holly	0008	24	BIT	holly3@foo.edu
Irene	0009	20	CS, Math	ireneg@foo.edu

- Where and how should I *store* “Big Data”?
  - Context? Retrieval vs analysis vs synthesis
  - Are sources homogeneous/centralized or heterogeneous/decentralized?
    - Keep my data organized as is or pay a conversion cost (one-time or recurring)?
  - Bring data closer to compute? Reduce in-flight transform overhead?
- How do I extract *insight* from “Big Data”?
  - How to *efficiently* analyze known properties from disparate sources? Task 1
  - How to *efficiently* synthesize a new property across a dataset? Task 2
- How do I design *scalable*, *future-ready* techniques to analyze “Big Data”?
  - How to utilize distributed and/or heterogeneous hardware?
    - Multi-node, multi-/many-core, streaming, tensor, or reconfigurable?
    - Memory, storage, network technologies and topologies?
  - Is there / could there be a *lingua franca* for data analysis?
    - *Open standards?* Competitive w/ off-the-shelf proprietary data system
    - How well do they leverage modern HPC and/or cloud hardware?



V3

Two-Pronged Approach

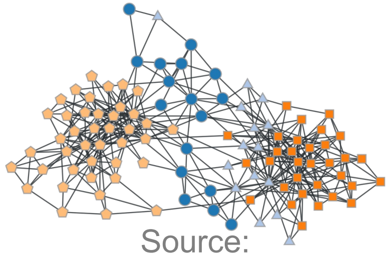
- Simplifying HPC data analytics workflows
  - Many HPC languages/frameworks exist, but are difficult to use for non-experts
  - Growing need for "behind-the-scenes" acceleration in large data analytics workflows
  - Enter Apache Arrow → end-to-end HPC data analytics framework
    - GPU acceleration
    - ML + graph analytics
    - Python API
  - How does the performance of Apache Arrow hold up on scientific workflows?
- Accelerating large-scale data analytics
  - Novel approaches/algorithms needed to accommodate growing datasets
  - Special area of interest: graph analytics
    - Wide variety of applications
    - Challenging to accelerate in a scalable manner due to sparse data & irregular memory access
    - Web-scale graphs (1+ billion edges) present memory bottlenecks
    - More accurate algorithms, more expensive
  - Exploring heuristics for accelerating large-scale graph analytics
    - Algorithmic refinements
    - Data reduction
    - Parallel and distributed computing

Proposed Tasks for V3-24

(Memberships Needed: Mandatory + Optional, e.g., 2+1)

- Task 1: Democratized High-Productivity Analytics (1+0)
    a) Construct workflows of interest to SHREC members
    b) Evaluate the performance, accuracy, and interoperability of Apache Arrow-supported workflow

- Task 2: Scalable Graph Analytics (1+1)
    a) Scale out our accelerated algorithm for graph clustering
    b) Perform algorithmic refinements to further accelerate graph clustering
    c) Explore the applications of OpenSHMEM and supernodes for data and workload distribution in parallel graph clustering



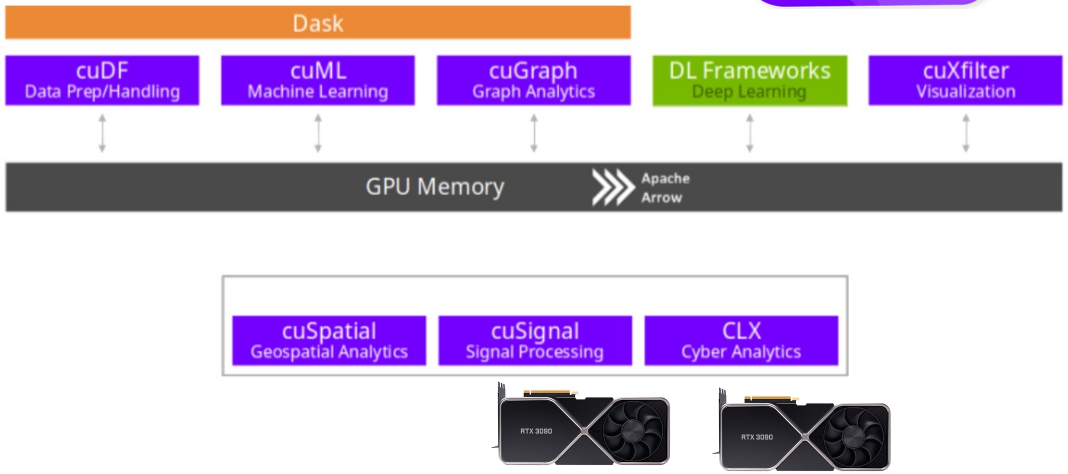
Source: *graph-tool.skewed.de*

Task 1: Democratized High-Productivity Analytics

V3

RAPIDS

- Motivation
  - Apache *Arrow*: Platform for building high-performance apps to analyze large-scale data
    - Idea? *Common* in-memory representation to streamline the format-thrashing common to analytics pipelines
    - Features
      - Improves performance of analytical algorithms, e.g., columnar layout enables vectorization (SIMD)
      - Improves efficiency of moving data between systems via Arrow *Flight*
- Approach
  - Evaluate the efficacy of Apache-integrated libraries in an appropriate environment (e.g., C++, Python atop C++ library, Ibis, R, Julia, and Apache Spark) – ***performance, accuracy, and interoperability***

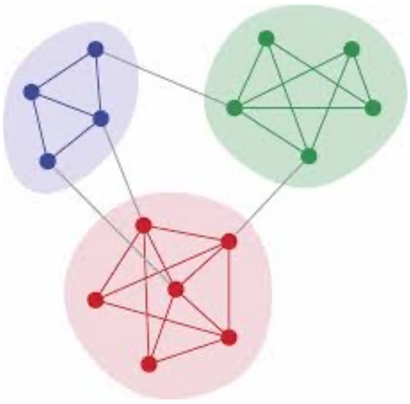


- Tasks 4a and 4b
  a) Construct proxy data analysis workflow(s) of interest to SHREC members
     e.g., SparkLeBLAST with Spark+Arrow, others?
  b) Evaluate the performance, accuracy, and interoperability of Apache Arrow-supported workflows and data engines

V3

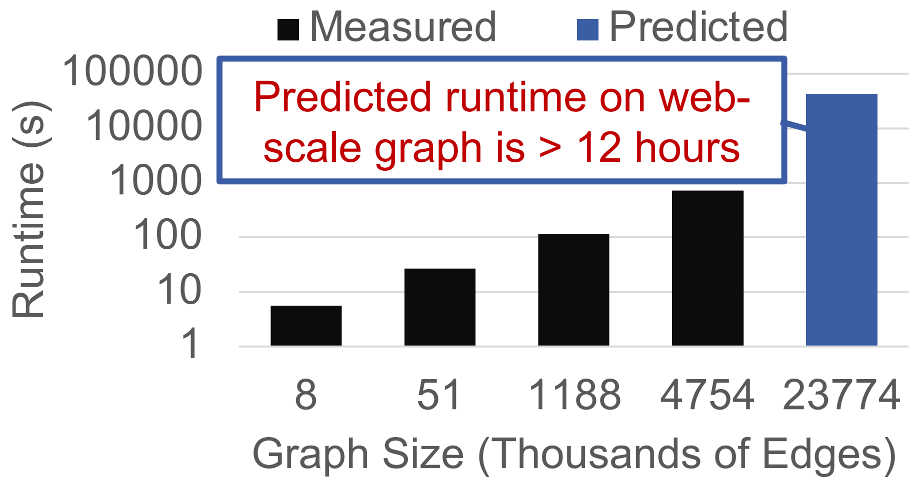
Task 2: Scalable Graph Analytics

- Motivation
  - Graph clustering: NP-hard graph analytics problem – grouping strongly connected vertices together
    - Clusters correspond to functional groups
    - Applications in many domains (e.g., bioinformatics, networking, finance, etc.)
  - Stochastic block partitioning (SBP): Accurate but slow graph clustering algorithm based on statistical inference
    - State-of-the-art implementation takes too long to process web-scale graphs (1+ billion edges)
- Approach
  - Accelerate SBP algorithm while maintaining accuracy



Source:
Nature Physics

State-of-the-Art
Accelerated SBP Runtime



Graph source: IEEE/Amazon/IEEE Graph Challenge
Hardware: 64 nodes with 128 cores and 256GB of RAM

Tasks 2a, 2b, 2c

a) Scale out our accelerated algorithm for graph clustering

b) Perform algorithmic refinements to further accelerate graph clustering

c) Explore the applications of OpenSHMEM and supernodes for data and workload distribution in parallel graph clustering

OpenSHMEM

Milestones, Deliverables, and Budget

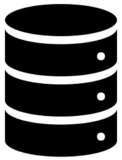
- **Major Milestones (Tasks: T1-T2)**
  - T1: Democratized High-Productivity Analytics: ***Proxy data workflows*** & e***valuation of data engines***
  - T2: Accelerated Graph Analytics: ***Novel accurate and scalable algorithms for graph clustering***
- **Deliverables**
  - Monthly progress reports, along with mid-year and end-of-year full reports
  - 1-2 publications at top-tier conference venues or journals

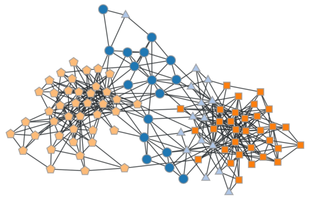
- **Recommended Budget**
  - Minimum: 2 memberships (100 votes)
  - Maximum: 3 memberships (150 votes)

Conclusion

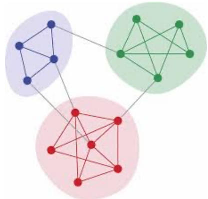
- Enable ***scalable*** analysis of ***increasingly large*** and varied real-world data sets:
  Tabular, Database, Graph, etc.

Name	ID	Age	Major	Email
Alice	0001	20	CS	alice@foo.edu
Bob	0002	23	ECE	bob23@foo.edu
Charles	0003	19	CS, ECE	charles1@foo.edu
Dana	0004	18	ECE	dana@foo.edu
Edward	0005	21	ECE	eddie2@foo.edu
Farah	0006	22	CS	farah@foo.edu
George	0007	19	CS	george@foo.edu
Holly	0008	24	BIT	holly3@foo.edu
Irene	0009	20	CS, Math	ireneg@foo.edu





- Implement and evaluate ***performance*** of ***distributed*** and ***heterogeneous*** solutions for data analytics:
  GPU-enabled tabular and database processing engines
  Multi-node graph clustering



Member Benefits

- Direct influence over algorithms, frameworks, and languages studied
- Direct benefit from new workflows, tools, datasets, codes, models, and insights created as well as new metrics of evaluation
- Direct insights from R&D and analysis of data processing engines

Appendix

Task 2: Scalable Graph Analytics

Scale out our distributed algorithm for graph clustering

Problem

- Our distributed graph clustering code has been successfully scaled to 64 nodes, processing a 194M edge graph in under an hour
- However, scaling to web-scale graphs (>1B edges) remains a challenge due to
  - High memory consumption, high communication overhead, Amdahl's law

Proposed Solutions

- Reduce memory consumption of distributed algorithm
  - Via sampling/supernodes, optimized data structures, etc.
- Reduce MPI communication overhead
  - Via one-sided and/or selective communication
- Reduce impact of Amdahl's law
  - Via asynchronous execution and algorithmic refactoring

Baseline: 0.5
Optional: 0

Task 2: Scalable Graph Analytics

Perform algorithmic refinements to accelerate graph clustering

Problem

- Even with distributed processing, graph clustering has an inherent scaling problem due to super linear runtime and high memory consumption in early iterations

Proposed Solutions

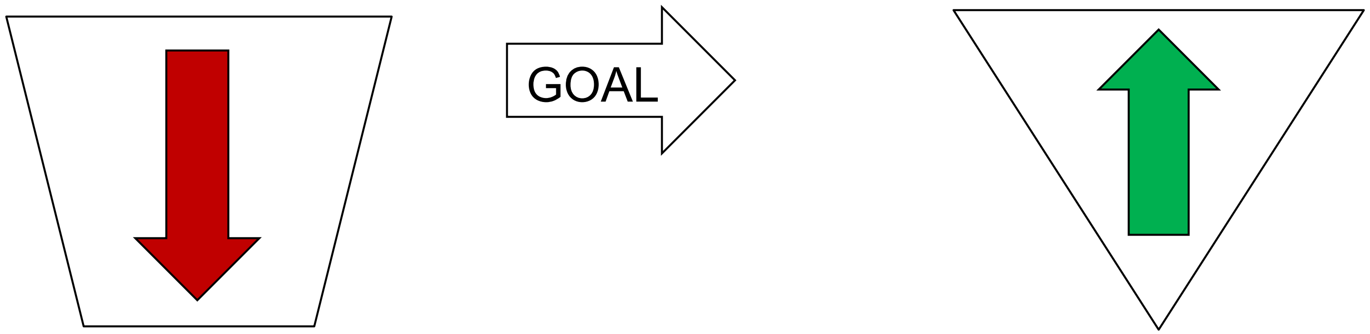
Contemporary Graph Clustering

Proposed Graph Clustering

Start with #clusters = #vertices

Split clusters until optimal # is found

- Large initial search space → very slow initial iterations
- Very high memory usage in early iterations



- Small initial search space → fast initial iterations
- Low memory usage in early iterations

Merge clusters until optimal # is found

Start with #clusters = 1

Baseline: 0.5
Optional: 0

Task 2: Scalable Graph Analytics

V3

Explore the applications of OpenSHMEM and supernodes for data and workload distribution in parallel graph clustering

Problem

- Currently, data distribution for distributed graph analytics is an open problem
  - In a sense, you need to cluster the graph in order to then again cluster the graph
  - The state-of-the-art distributed stochastic block partitioning implementation lacks data distribution → limited scalability
- Sampling, while reducing memory requirements, can negatively impact clustering accuracy

Proposed Solutions

- Explore OpenSHMEM as an avenue for data distribution
  - Potentially lower communication overhead due to elimination of all-to-all MPI communication in state-of-the-art
- Explore agglomerated supernodes as an alternative to sampling for data reduction
  - Less information loss, since number of edges remains constant in the whole and summarized graphs
- Explore "splitting" of high-degree supernodes to improve workload balance and dependency management in parallel graph clustering implementations

Baseline: 0
Optional: 1