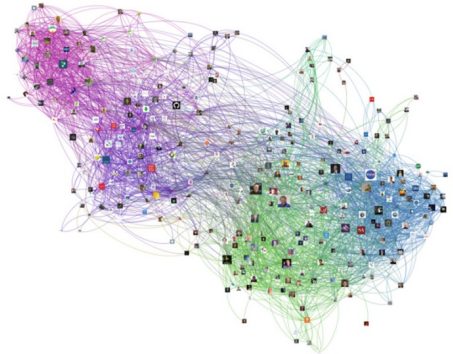
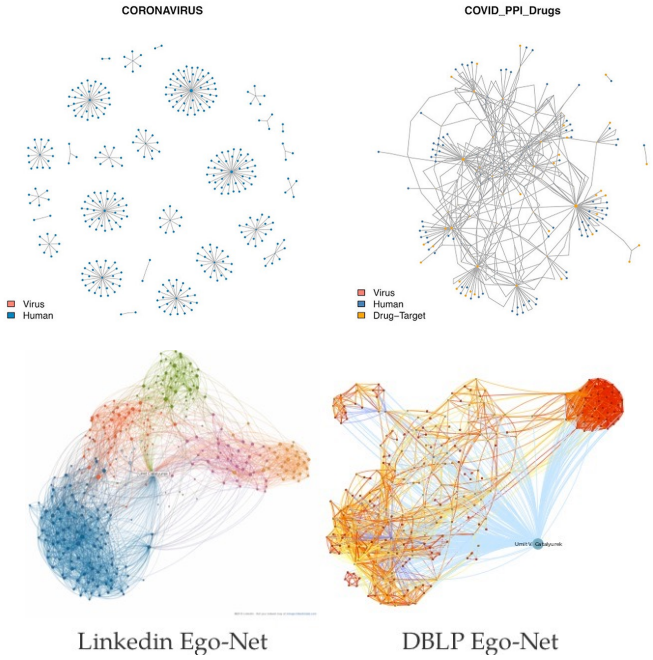
HPC Graph Analytics on the OneGraph Model

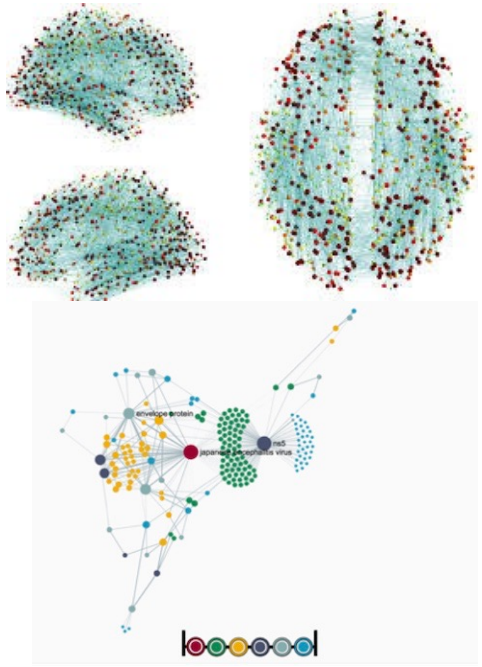
SIAM CSE 2023
March 3, 2023

Ümit V. Çatalyürek, Brad Bebee, Dave Bechberger, Willem Broekema, Hal Cooper, Olaf Hartig, Ankesh Khandelwal, Ora Lassila, Kelvin Lawrence, Carlos Manuel Lopez Enriquez, Michael Schmidt, Ronak Sharda, **Lefteris Sidirourgos**, Gabriel Tanase, Bryan Thompson, and Gregory Williams
Neptune Team, Amazon Web Services

Graphs are Ubiquitous







They are growing. Up to billions of vertices and edges

Fast, efficient analysis is important and pervasive

Many graph processing frameworks, and databases, have been proposed/developed

Image credits:

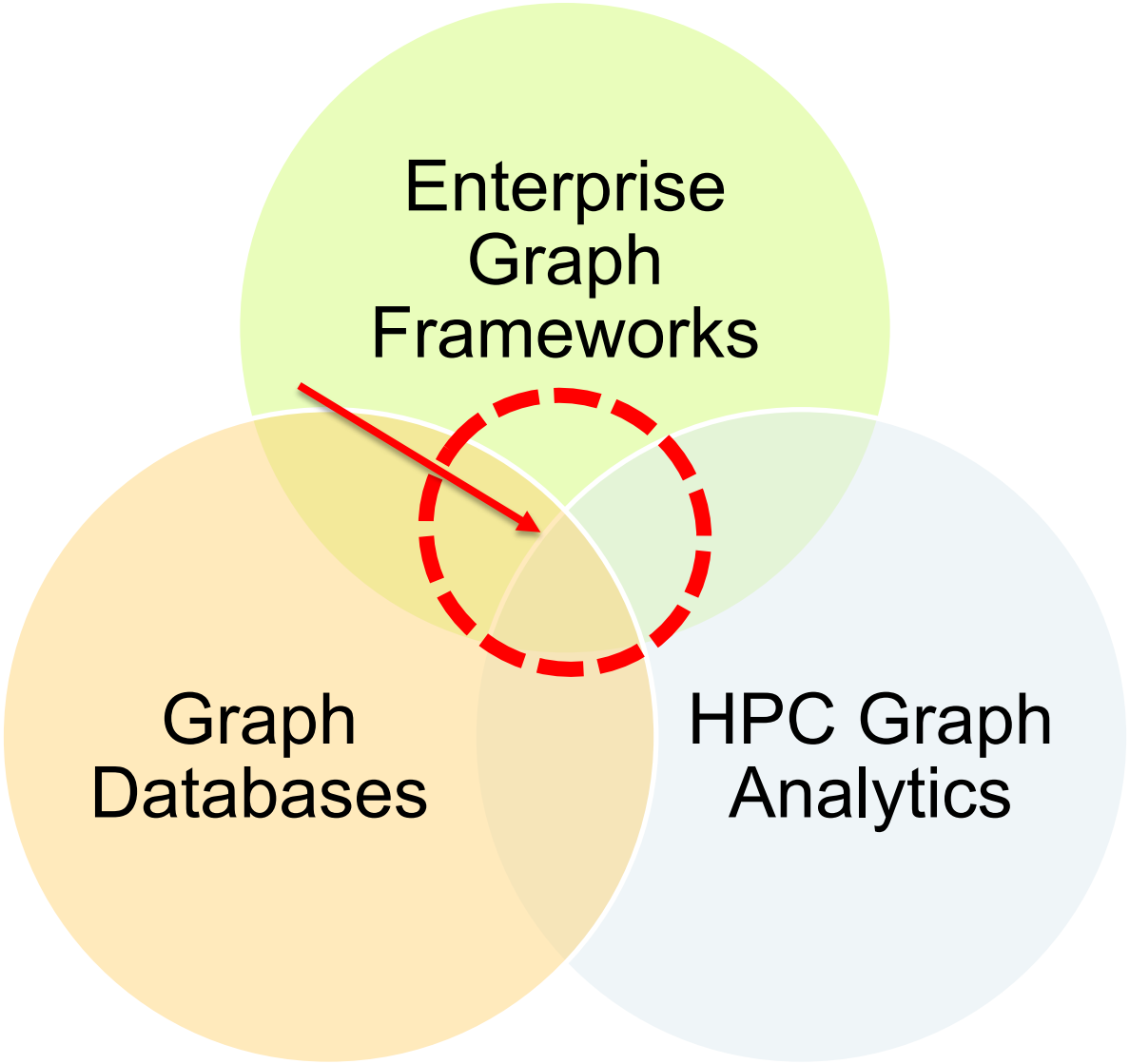
Jenn Caulfield, Social network vector illustration, 2018

Gerhard et al., Frontiers in Neuroinformatics 5(3), 2011

Albert-László Barabási/BarabasiLab 2019

Caleb Jonson, How to Visualize Your Twitter Network, 2014

Landscape of current “Graph World”



Scalability

Our goal is to have a performance that is in **small-constant factor** from HPC / State-of-the-art Graph Analytics, yet provide easy to maintain and productive development environment.

- F. McSherry, M. Isard, and D. G. Murray, “Scalability! But at what COST?” HotOS, 2015.
- N. Satish, N. Sundaram, M. M. A. Patwary, J. Seo, J. Park, M. A. Hassaan, S. Sengupta, Z. Yin, and P. Dubey, “Navigating the maze of graph analytics frameworks using massive graph datasets”. SIGMOD 2014.

Graph Data Models

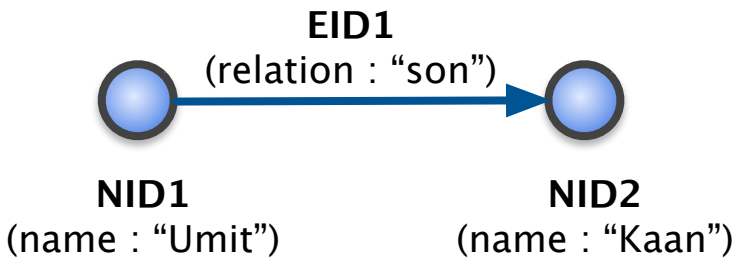
LPG: Labeled Property Graphs

Vertices

Nodes: Label/ID + *Properties* (set of key-value pairs)

Edges

Relationships: Label/ID + Type + Properties



RDF: Resources Description Framework

Vertices

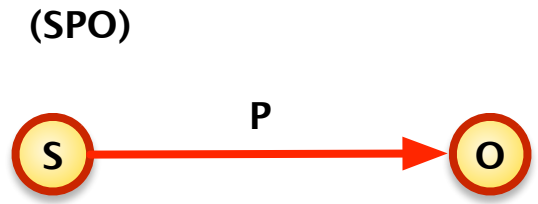
Resources: URIs

Attribute Values: Literals

Edges

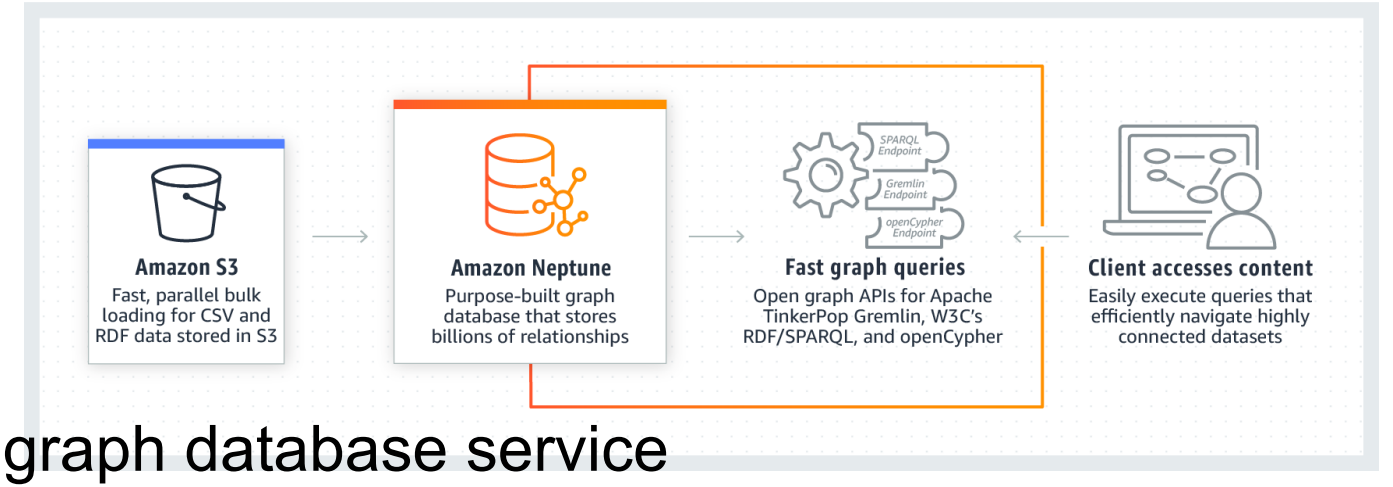
Relationships: URIs

RDF Triple: **S**ubject-**P**redicate-**O**bject



There is no internal structure for nodes and edges

Graph interoperability



- Amazon Neptune
  - managed, cloud-based graph database service
  - supports RDF (SPARQL) and LPG (Gremlin & openCypher)
- User has to choose either RDF or LPG
  - this choice also determines which query languages are available
  - the choice is not always easy, and is hard to reverse later
- RDF vs. LPG
  - RDF offers a formal model, LPG not so much
  - RDF is “sometimes seen as academic”, and developers tend to prefer LPG
  - different strengths and weaknesses

Graph interoperability

- What if we did not have to choose between RDF and LPG?
- What if we could use Gremlin over RDF, or SPARQL over LPG?
- Interoperability: single graph (meta)model, free use of any query language
  - we are not interested in “qualified” interoperability where one meta-model is *implemented* using the other
- RDF-star is a step towards having LPG features in RDF

- 1G model (“one graph to rule them all”)
  - "Graph? Yes! Which one? Help!", O. Lassila, M. Schmidt, B. Bebee, D. Bechberger, W. Broekema, A. Khandelwal, K. Lawrence, R. Sharda, B. Thompson, arXiv:2110.13348v1, 2021.

Interoperability challenges

- Edge properties, multiple edge instances, reification
- Triples vs. graph abstraction
- Datatype alignment
- Graph partitioning
- Graph merging, external identifiers
- Lack of formal foundation
- Update query semantics

The Neptune team seek support from the broader community to look into these issues

Storage Challenges: Interoperability

- Interoperability: serve both RDF and LPG
- 1G Graph Storage
  - Three kinds of relations
    - Dictionaries: URIs/Literals → IDs
    - Graph Structure: Topologies – relations between (S)ubject and (O)bject, in other words between “vertices”
    - Graph Data: Values – properties of vertices and edges
  - In 1G, Edges/Properties (of vertices and edges) can become “vertices”
- Relations are partitioned
  - 1D: Dictionaries, Vertex Properties etc.
  - 2D: Topology and properties of edges (collocated for performance)

Storage Challenges: Dynamic Partitioned Data

- Graph is not static (well, obviously!)
  - Many HPC Graph Analytics kernels assumes graph is not changing.
  - Even dynamic ones conveniently *ignores* deletion.
- How to (dynamically) distribute data?
- System generated IDs are uniform random
  - Notice that graph comes as vertices as URIs
  - Load-balanced partitioning (declustering) is favored against locality for initial load
  - Graph-aware re-partitioning can be done after graph is loaded
- Node partition (1D) vs Edge partition (fine-grain 2D) vs Blocked partition (coarse-grain 2D)
  - Blocked partition is used as a sweet spot between performance and architecture agnostic algorithm development [1]

[1] PGAbB: A Block-Based Graph Processing Framework for Heterogeneous Platforms
Abdurrahman Yasar, Sivasankaran Rajamanickam, Jonathan W. Berry, Umit V. Catalyurek
https://arxiv.org/abs/2209.04541

Storage Challenges: Scalability and Transactions

- Scalability: Scaling Up (vertical/single-node) and Scaling Out (horizontal/multi-node)
  - Read scaling is “easy”
  - Write scaling with transaction support is challenging:
    - Distributed in-memory graph storage with logging is still challenging to implement.

- What does it mean to provide Graph Analytics under transactional system?
  - Transaction aware reads
    - Index-driven vs Scan-based kernels and dynamic tradeoffs based on cardinality estimates
  - Dynamic creation of “Views” for multi-iteration algorithms

Computational Infrastructure Challenges

- Can we implement once, and run everywhere: from multi-core to multi-host with potentially accelerators?
- Yes!
  - Multi-Level Intermediate Representation (MLIR) for Graphs
  - “Coarse-grained” Labeled-Dataflow Execution
- Can we support both OLAP and OLTP graph data management?
- Yes!
  - Native Storage
  - Advanced Scan Kernels
  - State-of-the-art Transactional Model

Conclusions & Future Directions

- Graphs ubiquitous and market is growing extremely fast
  - "*By 2025, graph technologies will be used in 80% of the data and analytics innovations, up from 10% in 2021, facilitating rapid decision making across the enterprise*" Gartner "Market Guide: Graph Database Management Solutions", M. Adrian, A. Jaffri, D. Feinberg, 24 May 2021.
- HTAP (i.e., Hybrid OLTP and OLAP) solutions are needed!
  - Enterprise Graph Systems gives the *illusion* of read scaling, while failing in absolute performance, and write/update scaling (they just leave that to IO system)
  - HPC Graph Analytics codes/libraries, are one-off, focused on narrow set of kernels and fail to provide end-to-end solutions
  - Existing "Real" Graph Databases, provides either OLTP or OLAP, but fails to deliver both
- Interoperability is a big challenge!
  - SPARQL, Gremlin and OpenCypher queries for both OLTP and OLAP workloads
- Graph as a Service
- **It is exciting times for Graphs!**