HPC Graph Analytics on the OneGraph Model

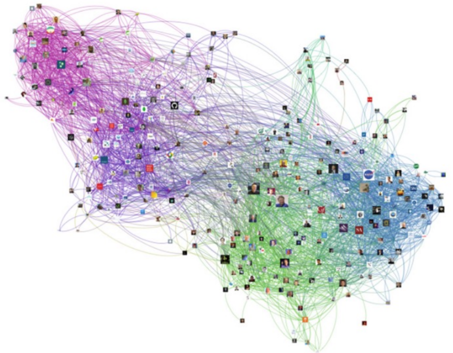
LDBC TUC 2023

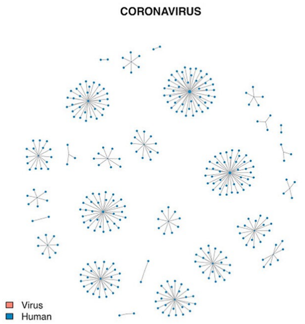
June 24, 2023

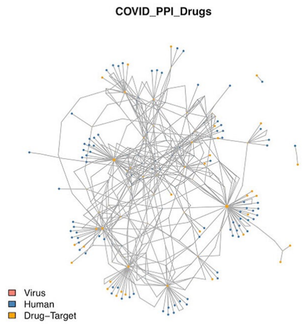
Ümit V. Çatalyürek
Amazon Scholar, Amazon Web Services
Professor, School of Computational Science and Engineering
Georgia Institute of Technology

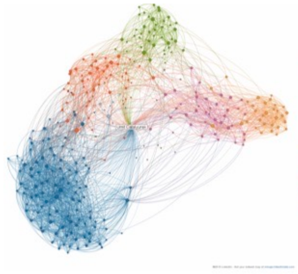
Brad Bebee, Dave Bechberger, Willem Broekema, Hal Cooper, Olaf Hartig, Ankesh Khandelwal, Ora Lassila, Kelvin Lawrence, Carlos Manuel Lopez Enriquez, Michael Schmidt, Ronak Sharda, Lefteris Sidirourgos, Gabriel Tanase, Bryan Thompson, and Gregory Williams
Neptune Team, Amazon Web Services

Graphs are Ubiquitous and Fun!

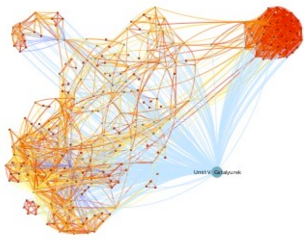




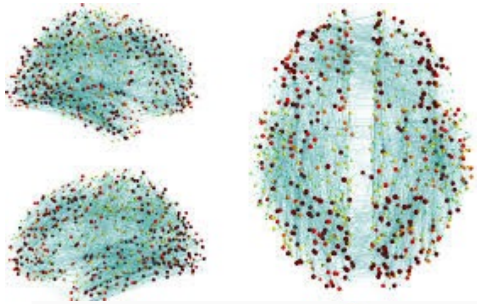




Linkedin Ego-Net



DBLP Ego-Net



They are growing. Up to billions of vertices and edges

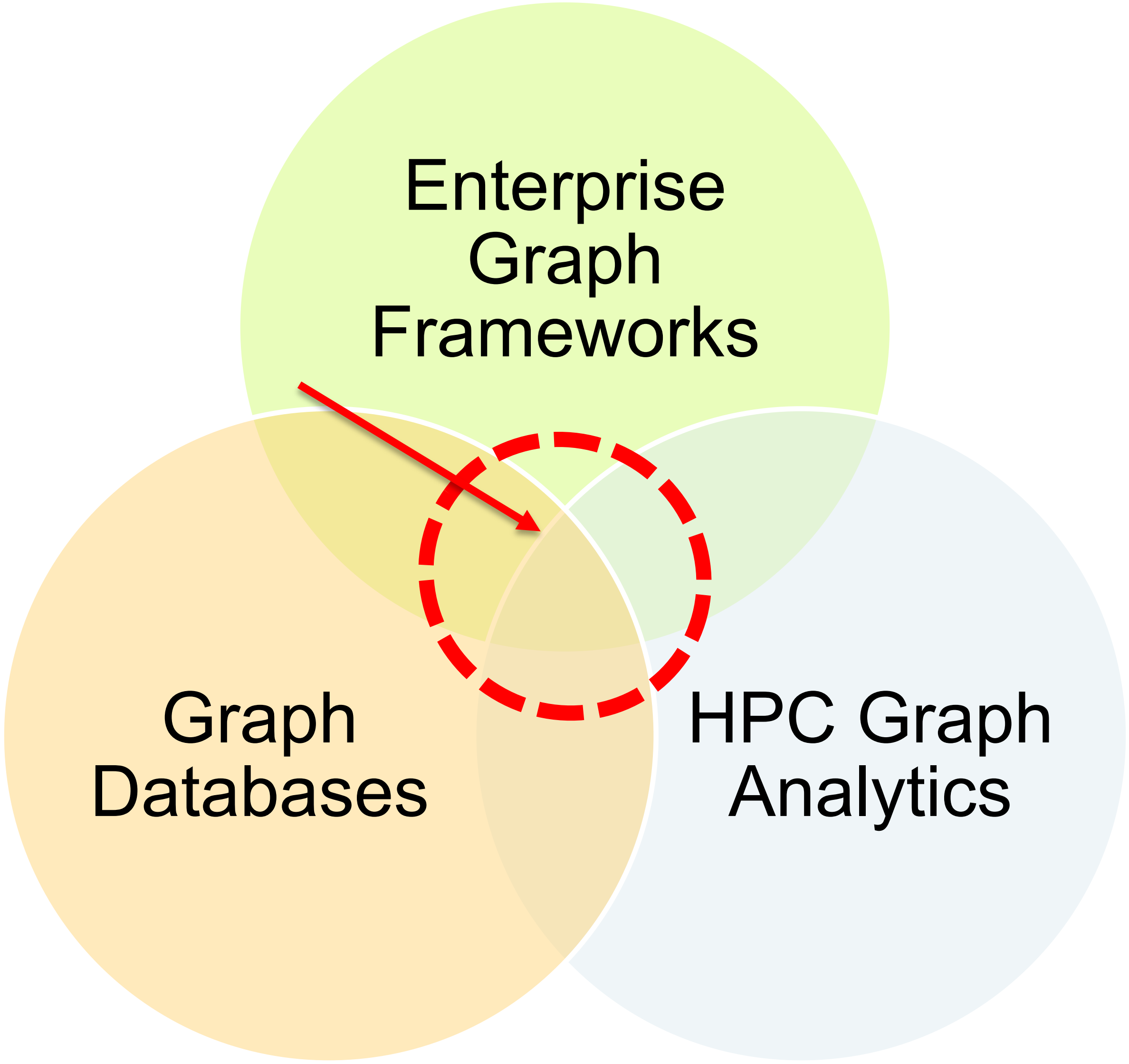
Fast, efficient analysis is important and pervasive

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

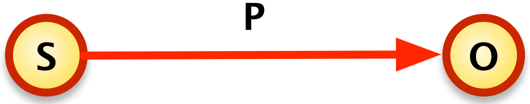
Attribute Values: Literals

Edges

Relationships: URIs

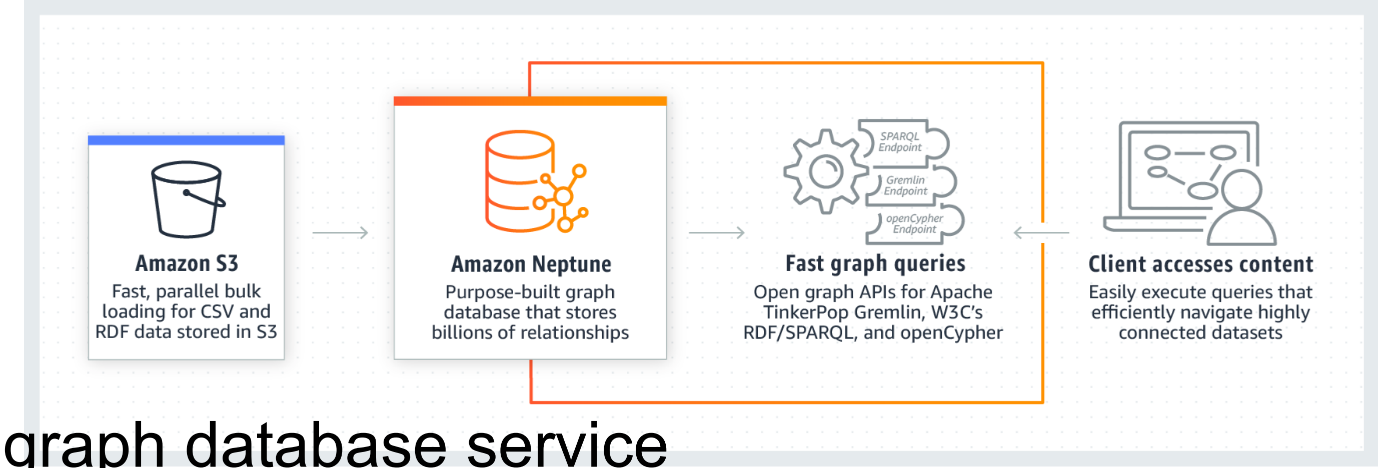
RDF Triple: **S**ubject-**P**redicate-**O**bject

(SPO)



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.

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 (sharded)
  - 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-ordering/re-labeling can be done after graph is loaded
- Sharding options: 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] (more on next slide)

[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

Why 2D?: PGAbB Results on Selected Graphs [1]

- Power9 (2 x 16 x 4) CPUs & Volta100 GPU.
  - 320 GB Host Memory. 32 GB Device Memory.
  - CPU-GPU bandwidth: ~60GB/s
- PGAbB: Kokkos at the backend with OpenMP (Host) and Cuda (Device)
- All wormalized wrt GAPBS

	Number of			
Graph	**Vertices**	**Edges**	**Triangles**	**CC**
Twitter7	41.6 M	1.2 B	34.8 B	0.001
Com-Orkut	3 M	117 M	627 M	0.041
Sk-2005	50.6 M	1.8 B	84.9 B	0.002
Kmer_V1r	214 M	232 M	49	0.000
Europe-OSM	50.9 M	54.1 M	61 K	0.003
Myciel.19	393 K	451 M	0	0
Kron-Scale21	2.1 M	91 M	8.8 B	0.044

		Social		Web	Gene	Road	Synthetic	
		twitter7	Orkut	sk-2005	kmer_V1r	eu_osm	myciel19	kron21
Galois	PR	0.83	1.01	1.01	0.89	1.03	6.96	0.78
	SV/LP	8.40	1.71	1.68	2.29	1.81	1.25	1.12
	CC	0.84	1.56	0.98	0.64	0.64	2.94	0.81
	BFS	0.26	0.59	0.46	0.34	2.14	0.39	0.18
	TC	0.69	1.06	0.63	0.90	1.21	0.44	0.40
Ligra	PR	0.39	0.60	0.99	0.43	0.53	2.59	0.72
	SV/LP	1.24	0.70	1.05	0.18	0.02	0.58	0.66
	CC	0.02	0.04	0.00	0.02	0.01	0.03	0.02
	BFS	0.61	0.67	0.93	0.68	0.16	1.37	0.82
	TC	0.31	0.35	0.12	0.30	0.17	0.43	0.69
LAGraph	PR	0.75	0.98	0.60	0.75	0.65	3.21	0.71
	SV/LP	14.24	1.64	0.89	0.30	0.13	7.70	0.92
	CC	0.17	0.21	0.12	0.14	0.05	0.27	0.09
	BFS	0.79	0.33	0.77	0.27	0.33	0.75	0.30
	TC	0.38	0.87	0.66	0.29	0.16	0.52	0.37
Galois-GPU	PR	0.00	2.72	0.00	1.01	1.49	12.12	1.62
	SV/LP	0.00	3.67	0.00	2.43	2.71	2.65	1.57
	CC	0.00	0.46	0.00	1.16	0.99	0.09	0.15
	BFS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	TC	1.03	0.85	0.90	0.00	0.00	0.38	0.65
Gunrock	PR	0.00	1.28	0.00	1.44	1.34	5.42	0.97
	SV/LP	0.00	1.88	0.00	3.18	1.22	3.90	0.97
	CC	0.00	0.24	0.00	1.51	0.44	0.14	0.09
	BFS	4.61	1.48	0.00	3.59	0.80	3.45	5.73
	TC	0.00	0.74	0.00	0.04	0.02	0.29	0.23
PGAbB-GPU	PR	4.20	4.72	0.74	0.53	0.64	13.60	2.30
	SV/LP	19.19	9.96	3.16	6.45	3.63	9.21	3.85
	CC	1.68	1.08	5.52	3.56	1.37	0.64	0.31
	BFS	0.18	0.85	0.97	0.28	0.32	1.06	0.27
	TC	3.09	3.39	2.34	0.52	0.32	2.87	2.33
PGAbB	PR	4.64	4.67	0.80	0.53	0.64	10.76	1.79
	SV/LP	18.02	5.95	1.90	5.73	2.95	7.70	1.98
	CC	1.25	1.53	2.14	1.91	0.96	2.40	0.87
	BFS	0.16	0.89	0.77	0.90	0.33	1.00	0.29
	TC	3.02	3.01	1.69	1.11	3.91	5.39	3.48

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

How about computational model

- Internally we provide
  - From “think like a vertex” to “think like a sub-graph (block/tile)”
  - Visitor model

- Externally:
  - Currently openCypher + with Graph API

- Ümit says
  - Low-hanging fruit: GraphBLAS

Storage Challenges: Scalability and Transactions

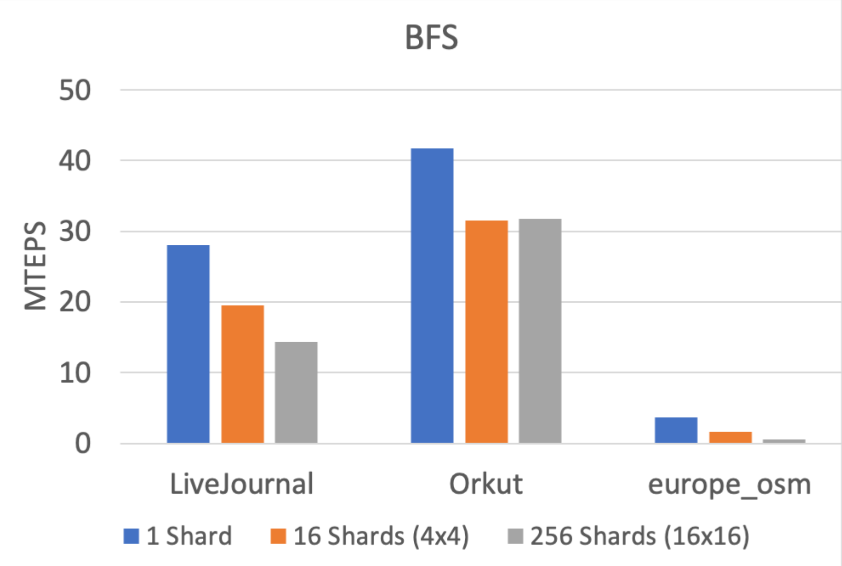
- Scalability: Scaling Up (vertical/single-node) and Scaling Out (horizontal/multi-node)
  - Read scaling is “easy”
  - Write scaling with transaction support is challenging:
    - Distributed in-memory graph storage with logging is still challenging to implement.

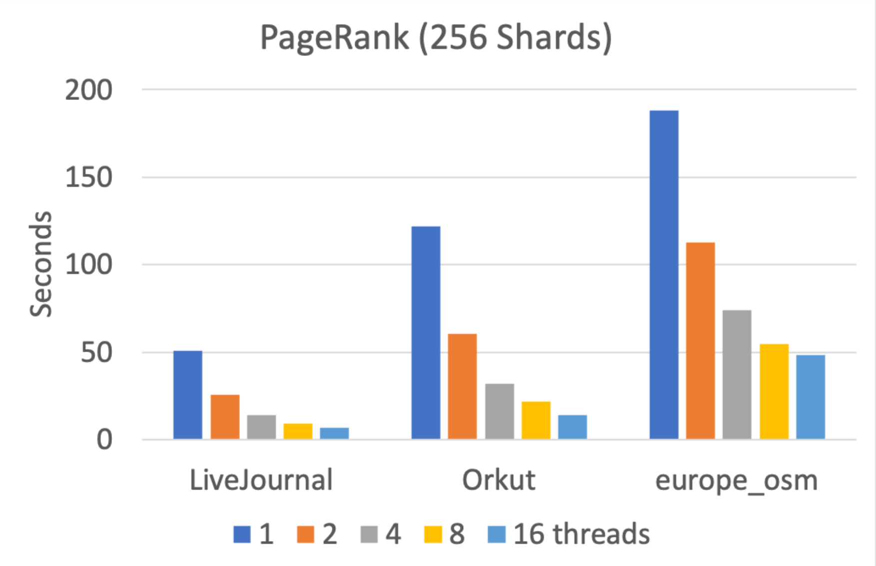
- What does it mean to provide Graph Analytics under transactional system?
  - Transaction aware reads
    - Index-driven vs Scan-based kernels and dynamic tradeoffs based on cardinality estimates
  - Dynamic creation of “Views” for multi-iteration algorithms

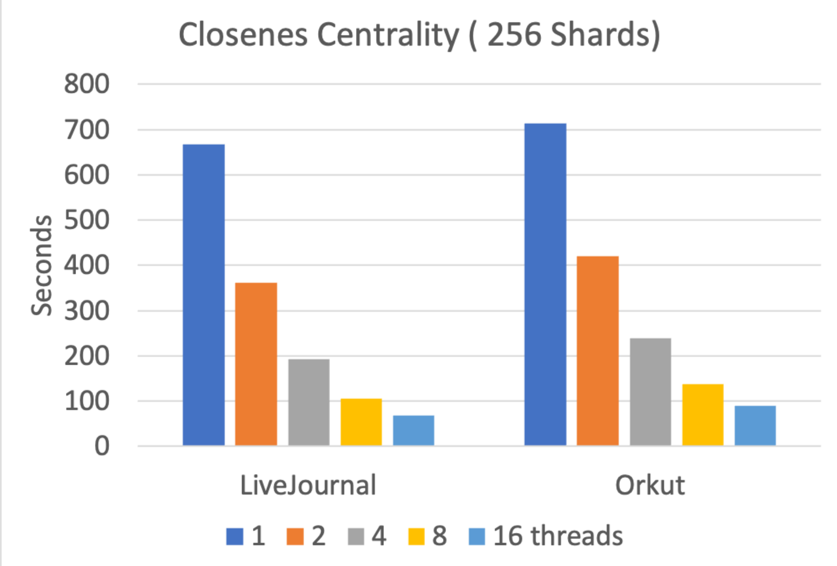
Effect of Shading to Performance

- **Early results** on a single, **old** EC2 instance
- Each shard executed sequentially (no fine-grain parallelism)
- Results show expected behavior:
  - Performance of BFS correlated with avg degree
  - 2D partitioning/sharding is not ideal for BFS, especially for very sparse data, but works well for almost all others, such as PageRank

name	#vertices	#edges (undirected)	avg degree
com-LiveJournal	3,997,962	34,681,189	17.35
com-Orkut	3,072,441	117,185,083	76.28
europe_osm	50,912,018	54,054,660	2.12







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 (MVCC, …)

Conclusions & Future Directions

- 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!**