Overview

• Motivation
• Options
• Evaluation
• Results
• Lessons Learned
• Moving Forward
Massive Streaming Semantic Graphs

Features
- Millions to billions of vertices and edges with rich semantic information (name, type, weight, time), possibly missing or inconsistent data
- Thousands to millions of updates per second
- Power-law degree distribution, sparse ($d(v) \sim O(1)$), low diameter

Financial
- NYSE processes >2TB daily, maintains 10’s of PB

Social
- 50,000+ Facebook Likes per second, 1.2B users
- 6,000 Twitter Tweets per second, 500M users

Google
- “Several dozen” 1PB data sets
- Knowledge Graph: 570M entities, 18B relationships

Business
- eBay: >17 trillion records, 5B new records per day
Given these graphs...

- **How do we store them?**
  - Memory vs. Disk
  - ACID vs. loose consistency
  - Single node vs. distributed
  - Semantic and temporal information
  - Numeric IDs

- **How do we query them?**
  - Simple neighbor queries
  - Traversal methods, filtering
  - Programming models vs. query languages

- **How do we compute meaningful answers?**
  - Algorithms and implementation paradigms
  - BSP, MapReduce, MPI, OpenMP, single threaded
  - C, C++, Python, Java, Scala, Javascript, SQL, Gremlin
**The Options**

<table>
<thead>
<tr>
<th>Owner / Maintainer</th>
<th>License</th>
<th>Platform</th>
<th>Language</th>
<th>Distribution</th>
<th>Cost</th>
<th>Transactional</th>
<th>Memory-Based</th>
<th>Disk-Based</th>
<th>Single-Node</th>
<th>Distributed</th>
<th>Graph Algorithms</th>
<th>Text-Based</th>
<th>Query Language</th>
<th>Embeddable</th>
<th>Software</th>
<th>Data Store</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>Oracle</td>
<td>x86</td>
<td>C/C++</td>
<td>Bin</td>
<td>Free</td>
<td>- X - X</td>
<td>X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>SQL</td>
<td>X - X - X</td>
<td>SQLDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>Proprietary</td>
<td>x86</td>
<td>C/C++</td>
<td>Bin</td>
<td>$180-$950</td>
<td>X - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>SQL</td>
<td>X - X - X</td>
<td>SQLDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQLite</td>
<td>Microsoft</td>
<td>x86-Win</td>
<td>C++</td>
<td>Bin</td>
<td>$898-$8952</td>
<td>X - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>SQL</td>
<td>X - X - X</td>
<td>SQLDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AllegroGraph</td>
<td>Franz, Inc.</td>
<td>x86</td>
<td>LikelyJava</td>
<td>Bin</td>
<td>Free-Ish/$$$</td>
<td>X - X - X</td>
<td>- X -</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>SPARQL,RDFS++</td>
<td>- X - X</td>
<td>GDB</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FlockDB</td>
<td>Twitter</td>
<td>x86</td>
<td>Java,Scala,Ruby</td>
<td>Src</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X - X</td>
<td>X - X</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphBase</td>
<td>FactNexus</td>
<td>x86</td>
<td>Java</td>
<td>Bin</td>
<td>Free, $15/mo, $20,000</td>
<td>X - X - X</td>
<td>- X -</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Bounds</td>
<td>X - X</td>
<td>X - X</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HyperGraphDB</td>
<td>Kobrix Software</td>
<td>x86</td>
<td>Java</td>
<td>Src</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>MVCC</td>
<td>X - X - X</td>
<td>HyperGDB</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InfiniteGraph</td>
<td>Objectivity</td>
<td>x86</td>
<td>Java/C++</td>
<td>Bin</td>
<td>Free-Trial/$5,000</td>
<td>Both - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Gremlin</td>
<td>X - X</td>
<td>X - X</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neo4j</td>
<td>Neo Technology</td>
<td>x86</td>
<td>Java</td>
<td>Src/Bin</td>
<td>Free-$6,000-$24,000</td>
<td>X - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OrientDB</td>
<td>NuvolaBase Ltd</td>
<td>x86</td>
<td>Java</td>
<td>Src/Bin</td>
<td>Free- Support</td>
<td>Both - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Titan</td>
<td>Aurelius</td>
<td>x86</td>
<td>Java</td>
<td>Src/Bin</td>
<td>Free- Support</td>
<td>Both - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bagel</td>
<td>UC Berkeley</td>
<td>x86</td>
<td>Java,Scala,Spark</td>
<td>Src</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BGL</td>
<td>Boost / IU</td>
<td>x86 / C++</td>
<td>C++</td>
<td>Src/Bin</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Faunus</td>
<td>Aurelius</td>
<td>x86</td>
<td>Java</td>
<td>Src/Bin</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Giraph</td>
<td>Apache</td>
<td>x86</td>
<td>Java</td>
<td>Src</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphStream</td>
<td>University Le Havre</td>
<td>x86</td>
<td>Java</td>
<td>Src/Bin</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hama</td>
<td>Apache</td>
<td>x86</td>
<td>Java</td>
<td>Src</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTGL</td>
<td>Senda NL</td>
<td>x86</td>
<td>C++</td>
<td>Src</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetworkX</td>
<td>Los Alamos NL</td>
<td>x86</td>
<td>Python</td>
<td>Src/Bin</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEGASUS</td>
<td>CMU</td>
<td>x86</td>
<td>Java</td>
<td>Src/Bin</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STINGER</td>
<td>GT / GTRI</td>
<td>x86 / XMT</td>
<td>C</td>
<td>Src</td>
<td>Free</td>
<td>- - X - X</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>Cypher</td>
<td>X - X</td>
<td>GDB/NoSQL</td>
<td>X - X - X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uriska</td>
<td>Yarc Data / Cray</td>
<td>Proprietary</td>
<td>XMT</td>
<td>Likely C++</td>
<td>Bin</td>
<td>$$ - $$</td>
<td>- X - X</td>
<td>- X -</td>
<td>-</td>
<td>-</td>
<td>SPARQL</td>
<td>- X - X</td>
<td>Appliance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* see page 2 of the paper
* DEX is now Sparksee
Evaluation

• **Four** fundamental graph kernels
  – Prefer a particular implementation
  – **Emphasize common programming / traversal styles**
  – Measure performance in multiple ways
    • **Time** to completion
    • **Memory** in use (MemoryMXBean for Java, OS reporting otherwise)
    • Qualitative analysis of **capabilities** and development experience
  – If not provided by package, written by naïve programmer

• **Same R-MAT graphs used with all packages**
  – Vertices: 1K (tiny), 32K (small), 1M (medium), and 16M (large)
  – Edges: 8K (tiny), 256K (small), 8M (medium), 128M (large)
  – Graphs of 1B edges considered, but not included due to number of systems that couldn’t work with large
Evaluation

• Single Source Shortest Paths (SSSP)
  – Must be a breadth-first traversal of the graph
  – Level-synchronous parallel where possible
  – Output unweighted distances from source to all vertices

• Application
  – Used for routing and connectivity
  – Building block for other algorithms (e.g. betweenness centrality)
  – Graph 500 benchmark
Evaluation

• Connected Components
  – Based on Shiloach-Vishkin
  – Edge-parallel label-pushing style
  – Global graph metric that touches all edges in the graph

• Properties
  – Not theoretically work efficient, but edge-parallel pattern is representative of a broad class of graph algorithms
  – Read-heavy with sparse writes
  – Cache friendly graph traversal with cache-unfriendly random label read / assignment
Evaluation

• PageRank centrality algorithm
  – Vertex-parallel Bulk Synchronous Parallel (BSP) power-iteration style
  – Measure of influence over information flow and importance in the network
  – Representative of a broad class of iterative graph algorithms

• Update benchmark
  – Perform edge insertions and deletions in parallel
  – Random access and modification of the structure
  – Real-world networks are in constant motion as new edges and vertices enter the graph
DISCLAIMERS

• We are not expert Java programmers, Hadoop tuners/magicians, Python gurus (maybe a few), Scala wizards

• We are some of the primary maintainers of the STINGER graph package

• The code for these implementations is available at github.com/robmccoll/graphdb-testing
  – Test code is BSD, package licenses vary
  – We apologize if you feel that your particular work has been slighted or misrepresented
  – If any of these packages belong to you, we invite and encourage you to improve on our results and submit them back to the repository
## Maximum Size Completed

<table>
<thead>
<tr>
<th>Package</th>
<th>Tiny</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>boost</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>dex</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mtgl</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>neo4j</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>networkx</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>orientdb</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>sqlite</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stinger</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>titan</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>bagel</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pegasus</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>giraph</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Determined by crashing, running out of memory, or running an algorithm for longer than 24 hours.

Exception: Dex had a 1 million element ($|V| + |E|$) limit.
Memory Usage

3 orders of magnitude difference in memory usage

![Graph showing memory usage for different graph packages](image)

- **Motivation**
  - Options
  - Evaluation
  - Results
- **Observation**
- **Moving On**
Page Rank Performance

**Small**
Range 10,000x

**Medium (x32)**
Range 10,000x
40~90x slower
Except Pegasus, Giraph

**Large (x1024)**
Range 100x
~20x slower
Except Pegasus
SSSP and Components
Small vs. Medium

Single Source Shortest Path – Small (|V|: 32K, |E|: 256K)

Connected Components – Small (|V|: 32K, |E|: 256K)

Single Source Shortest Path – Medium (|V|: 1M, |E|: 8M)

Connected Components – Medium (|V|: 1M, |E|: 8M)
Observations on Quantitative

- Big performance gaps, possible groups
  - High Performance (ms response)
  - Medium Performance (sub minute response)
  - Storage Performance
    - Perhaps suited for storage, but not analysis
    - Not language or system, just not written with graph algorithms in mind

- Some parallel, but many not
  - Graph libraries generally parallel
  - Not much parallelism in (single node) disk-backed DBs
    - Parallelism only increases random read and write
    - Bottleneck is disk bandwidth and latency

- Data scaling is hard to predict
- Full results available at:
  - github.com/robmccoll/graphdb-testing
  - arxiv.org/abs/1309.2675
Qualitative Observation

- Multitude of input file formats supported
  - No universal support
  - XML, JSON, CSV, or other proprietary binary and plain-text formats
  - Trade-off between size, descriptiveness, and flexibility
  - Challenge for data exchange
  - Edge-based, delimited, and self-describing easily parsed in parallel, translated to other formats

- Same data, same algorithms, same HW, DB / library is the difference

- No consensus on ACID / Transactions
  - No clear result on its affect on speed
  - Not clear that the applications exist

- No consensus on query languages, models
Moving Forward

• To advance HPC outside of our community, software must be accessible

• Some applications don’t compile out of the box
  – Self-contained packages are easiest
  – Package management systems don’t work well on nodes inside a cluster
  – Build problems drive users away

• Lack of consistent approaches to documentation
  – Well-written tutorials better than sparse full API doc
  – Lack of adequate usage examples
    • Inserting vertices and edges, querying for neighbors is not enough
    • What is the best way to traverse the structure?
    • How should data ingest, analysis, and extracting results be performed?

  – Highlights
    • NetworkX is very well documented, both examples and API doc
    • Titan’s API doc is very complete, but lacks examples of real-world usage
    • Bagel, STINGER lack formal documentation or have fragmented and occasionally inaccurate documentation
Acknowledgment of Support