Using Graphs for Unstructured Data

Keshav Pingali
CS, ECE and Oden Institute
The University of Texas at Austin
Intelligent Software Systems Group

• 12 PhD students and postdocs

• Funding
  • 3 DARPA projects, several NSF projects, companies

• Major projects
  • Galois system for parallel programming of unstructured problems
  • Adaptive control systems for principled accuracy/energy tradeoff
  • Using machine learning in systems software
Why graphs?

- Model for relationships between entities
  - Entity: graph vertex
  - Relation between entities: graph edge
- Example: web graphs like wdc12
  - 3 billion vertices
  - 128 billion edges
  - Average degree of vertex is ~42
  - > 1 TB on disk
- Big data sets are usually sparse graphs
  - Entity interacts with few other entities
  - Consequence of locality of interaction in physical world and cyberworld
## Graphs are Ubiquitous

<table>
<thead>
<tr>
<th>Machine Learning</th>
<th>Network analysis</th>
<th>Engineering design and simulation</th>
<th>Security</th>
<th>IoT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product recommendation systems</td>
<td>Web search engines Gene network mining</td>
<td>Finite-elements Non-equilibrium thermodynamics ASIC/FPGA design tools for synthesis, routing, placement and timing analysis</td>
<td>Real-time intrusion detection in computer networks Fraud detection Threat detection</td>
<td>Event hub data stores</td>
</tr>
</tbody>
</table>
Example: Word2Vec (ML application)

- **Word2Vec**: given large corpus of text, find word embeddings in vector space
  - Vector space model captures similarities between words
  - Applications in NLP, advertising, machine translation,…

- **Challenges**:
  - Training neural network takes days for many data-sets

- **Microsoft Research used Galois to reduce training time from days to hours**
Example: Security

- Finding bad actors in social networks
  - Centrality computations in large social network graphs
- Real-time intrusion detection in computer networks
  - Maintain evolving graph showing users’ activity and find blacklisted patterns
  - BAE used Galois for real-time intrusion detection
Example: Simulation, modeling and graphics

- Irregular meshes used in finite-elements and graphics
  - Mesh generation, refinement, coarsening, partitioning are graph computations
- Nonequilibrium thermodynamics: Boltzmann transport
  - Sweeps codes
- 3D mesh generation in Galois for pollution modeling in Spain
How to Think About Graph Applications
Graph Problem Classification

- **Generation**: Morph graph structure by adding/removing nodes/edges
  - Finite elements: Ansys, Boeing, Fluent, HeartFlow
  - EDA tools: Cadence, Intel, Synopsys, Xilinx
- **Labeling**: Compute node/edge attributes (graph structure is invariant)
  - Graph analytics: search engines, network analysis
  - Recommendation systems
- **Querying**: Path or structural queries in graphs (graph is invariant)
  - Security (intrusion detection): BAE, Carbon Black
  - Graph Databases: Neo4j, TigerGraph, RedisGraph
- **Mining**: Find patterns (motifs) in graph
  - Network analysis: 23andMe, Ancestry.com, intelligence agencies
  - Marketing: Amazon, Walmart
Graph Application Classification

Problems
- Generation
- Labeling
- Querying
- Mining

Graphs
- Type
  - Static
  - Streaming
  - Dynamic
- Diameter
  - Low
  - High

Algorithms
- Task-parallel
- Data-parallel
- Asynchronous
- Round-based
- General operator
- Vertex operator

And

Or
What Galois Supports

- Galois programs are C++ programs w/patterns
- Runs on CPUs and GPUs
- Subset of programming model supported on clusters
- Supports applications for all graph problems (see next slide)
Galois Ecosystem

**APIs**
- Pangolin: Graph mining
- GraphBLAS: Graph analytics
- OpenCypher: Graph querying

**Galois**
- Galois graph engine:
  - C++ programs
  - Data structure library
  - Runtime

**Architectures**
- CPUs, GPUs, clusters
Performance Studies
Graph generation: Word2Vec training (W2V, GEN vs. Galois)

- Implementations:
  - **W2V** (Word2Vec) and **GEN** (Gensim): state-of-the-art shared memory implementations
  - **GW2V**: Distributed Word2Vec Implemented on top of Galois

- **CPU cluster**: Microsoft Azure
  32 machines, each m/c has Intel Xeon E5-2667 CPUs (16 cores)

- **Take-away**: training time reduced from days to hours
Graph analytics: clusters (Galois)

clueweb12 (~50B edges)

**KNL cluster: TACC Stampede 1**
128 machines, each m/c has KNL CPUs (96 cores)

**CPU cluster: TACC Stampede 2**
128 machines, each m/c has 2 Skylake CPUs (24 cores)

**GPU cluster: Bridges@PSC**
32 machines, each m/c has 2 NVIDIA Tesla P100 GPUs
Graph Analytics: Single machine + Optane (GAP, Ligra vs. Galois)

GAP: Parallel library (Berkeley)
Ligra: Vertex program DSL (CMU)
Graph: clueweb12 (~1B vertices, ~50B edges, web crawl)
Machine: Intel Xeon Cascade Lake with 48 cores (x2 hyperthreading)
Take-away: Galois programs are 2-4X faster than GAP and Ligra programs
Graph Mining: (RStream, Arabesque, Kaleido vs. Pangolin/ Galois

- Applications: CF5: 5 cliques, MC3: motif counting, FSM: frequent subgraph mining
- Pangolin: API implemented on top of Galois
- Machine: Intel Xeon Gold (Skylake) with 28 cores
- Take-away: >50X faster than RStream and Arabesque, 3X faster than Kaleido
- GPU implementation of Pangolin: ~14X faster than CPU Pangolin
Summary

• Graphs are everywhere
• Graph algorithms are diverse
• Galois
  • supports full range of graph applications such as generation and mining without compromising on performance for simpler applications like graph analytics
  • easy to implement APIs for particular graph problems on top of system
  • used by several companies and academic groups