Understanding Trolls with Efficient Analytics of Large Graphs in Neo4j

David Allen, Amy E. Hodler, Michael Hunger, Martin Knobloch, William Lyon, Mark Needham, Hannes Voigt

BTW Rostock
Feb 2019
Michael Hunger

Director Neo4j Labs at Neo4j
@mesirii | michael@neo4j.com
Agenda

1. Graph Databases vs. Graph Processing
2. Neo4j Graph Platform
3. Neo4j Graph Algorithms
4. Application in SNA on Twitter Troll Dataset
Why graphs?
The world is a graph – everything is connected

- people, places, events
- companies, markets
- countries, history, politics
- sciences, art, teaching
- technology, networks, machines, applications, users
- software, code, dependencies, architecture, deployments
- criminals, fraudsters and their behavior
What are people using Neo4j for?
Neo4j - Transforming 100s of Large Enterprises
For Over 14 Years

Real-time promotion recommendations
- Record “Cyber Monday” sales
- About 35M daily transactions
- Each transaction is 3-22 hops
- Queries executed in 4ms or less
- Replaced IBM Websphere commerce

Marriott’s Real-time Pricing Engine
- 300M pricing operations per day
- 10x transaction throughput on half the hardware compared to Oracle
- Replaced Oracle database

Handling Package Routing in Real-Time
- Large postal service with over 500k employees
- Neo4j routes 7M+ packages daily at peak, with peaks of 5,000+ routing operations per second.
Use Cases

Internal Applications
Master Data Management
Network and IT Operations
Fraud Detection

Customer-Facing Applications
Real-Time Recommendations
Graph-Based Search
Identity and Access Management
The labeled property graph model
Property Graph Model Components

Nodes

• Represent the objects in the graph
• Can be *labeled*
Property Graph Model Components

Nodes
• Represent the objects in the graph
• Can be labeled

Relationships
• Relate nodes by type and direction
Property Graph Model Components

**Nodes**
- Represent the objects in the graph
- Can be *labeled*

**Relationships**
- Relate nodes by *type* and *direction*

**Properties**
- Name-value pairs that can go on nodes and relationships.
Summary of the graph building blocks

- **Nodes** - Entities and complex value types
- **Relationships** - Connect entities and structure domain
- **Properties** - Entity attributes, relationship qualities, metadata
- **Labels** - Group nodes by role
Neo4j is a Graph Platform
Neo4j is a database

- ACID Transactions
- reliable
- fast
- 2-4 M ops/s per core
- binary & http protocol
- Clustering Scale & HA
- official Drivers
- no size limit
Neo4j is a graph platform
Graph Querying
Cypher

A pattern matching query language made for graphs

- Declarative
- Expressive
- Pattern Matching

Formal specification, SIGMOD paper:

Cypher: Express Graph Patterns

(:Person { name: "Dan"}) -[:LOVES]-> (:Person { name: "Ann"})
Cypher: CREATE Graph Patterns

```
CREATE (:Person { name:"Dan"}) -[:LOVES]-> (:Person { name:"Ann"})
```
Cypher: MATCH Graph Patterns

MATCH (:Person { name: "Dan"}) -[:LOVES]-> (whom)
RETURN whom
Cypher: Query Planner
Cypher: Query Plan

- different planners
  - e.g. IDP planner
- different runtimes
  - e.g. bytecode compiled
openCypher / GQL

- open source graph query language specification and reference implementation
- Multi-Vendor effort to standardize a Graph Query Language, see: [gqlstandards.org](http://gqlstandards.org)

GQL is a proposed new international standard language for property graph querying. The idea of a standalone graph query language to complement SQL was raised by ISO SC32/ WG3 members in early 2017, and is echoed in the GQL manifesto of May 2018.

GQL supporters aim to develop a next-generation declarative graph query language that builds on the foundations of SQL and integrates proven ideas from the existing openCypher, PGQL, and G-CORE languages.

GQL will incorporate this prior work, as part of an expanded set of features including regular path queries, graph compositional queries (enabling views) and schema support.
A graph query example
A social recommendation

Sushi restaurants in New York, New York that my friends like

IS_FRIEND_OF

LIKES

name: Zushi Zam

cuisine: Sushi

SERVES

IS_FRIEND_OF

LIKES

name: iSushi

SERVES

LOCATED_IN

location: New York

LOCATED_IN
MATCH (person:Person)-[:IS_FRIEND_OF]->(friend),
  (friend)-[:LIKES]->(restaurant),
  (restaurant)-[:LOCATED_IN]->(loc:Location),
  (restaurant)-[:SERVES]->(type:Cuisine)
WHERE person.name = 'Philip'
AND loc.location='New York'
AND type.cuisine='Sushi'
RETURN restaurant.name
A social recommendation
Graph Algorithms
Source: John Swain - Twitter Analytics Right Relevance Talk
Many Moving Parts!

Python Tweet Collection (includes user data) → Rabbit MQ → MongoDB → Neo4j → R Scripts (Graph Stats, Community Detection) → MySQL → Tableau

Moved from Twitter Search API to Streaming API

Replaced Python Twitter libraries (Tweepy) with raw API calls

Streaming tweets in message queue
Full tweets and user data stored in MongoDB
Built graph for analysis in Neo4j from tweets persisted in MongoDB
Analysis in R iGraph libraries for algorithms
Some text analysis e.g. LDA topics
Results published in MySQL for Tableau
Graphml for import to Gephi with stats precalculated

Example Workflow Pipeline
Our Goal

Twitter Streaming API

Python Tweet Collection (includes user data) → Rabbit MQ → MongoDB → Neo4j

- R Scripts
  - Graph Stats
  - Community Detection

MySQL

Tableau

Graph Visualization

Example Workflow Pipeline
Neo4j Native Graph Database

Cypher Query Language

Analytics Integrations

Wide Range of APOC Procedures

Optimized Graph Algorithms
Finds the optimal path or evaluates route availability and quality

Determines the importance of distinct nodes in the network

Evaluates how a group is clustered or partitioned
Usage

1. Call as Cypher procedure
2. Pass in specification (Label, Prop, Query) and configuration
3. stream variant returns (a lot) of results

```
CALL algo.<name>.stream('Label','TYPE',{conf})
YIELD nodeId, score
```

4. non-stream variant writes results to graph returns statistics

```
CALL algo.<name>('Label','TYPE',{conf})
```
Cypher Projection

Pass in Cypher statement for node- and relationship-lists.

CALL algo.<name>(
    'MATCH ... RETURN id(n)',
    'MATCH (n)-->(m)
    RETURN id(n) as source,
    id(m) as target',  {graph:'cypher'})
Design Considerations

• Ease of Use – Call as Procedures
• Parallelize everything: load, compute, write
• Efficiency: Use direct access, efficient datastructures, provide high-level API
• Scale to billions of nodes and relationships
  Use up to hundreds of CPUs and Terabytes of RAM
1. Load Data in parallel from Neo4j
2. Store in efficient data structures
3. Run Graph Algorithm in parallel using Graph API
4. Write data back in parallel
Scale: 144 CPU
Neo4j Graph Platform with Neo4j Algorithms vs. Apache Spark’s GraphX

Neo4j provides same order of magnitude performance

Twitter 2010 Dataset
- 1.47 Billion Relationships
- 41.65 Million Nodes

Spark GraphX results publicly available
- Amazon EC2 cluster running 64-bit Linux
- 128 CPUs with 68 GB of memory, 2 hard disks

Neo4j Configuration
- Physical machine running 64-bit Linux
- 128 CPUs with 55 GB RAM, SSDs
Compute At Scale – Payment Graph

3,000,000,000 nodes and 18,000,000,000 relationships (600G)
PageRank (20 iterations) on 1 machine, 20 threads, 700G RAM

call algo.pageRank('Account','SENT',{graph:'big', iterations:20,write:false});

+------------------------------------------------------+
| nodes      | iterations | loadMillis | computeMillis |
+------------------------------------------------------+
| 3000000096 | 20         | 0          | 9845756       |
+------------------------------------------------------+

1 row
9845794 ms -> 2h 44m
# Evaluation

<table>
<thead>
<tr>
<th>Graph</th>
<th>#Nodes [M]</th>
<th>#Relationships [M]</th>
<th>Avg. out degree</th>
<th>Disk size [GB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pokec</td>
<td>(PK) 1.63</td>
<td>30.62</td>
<td>18.75</td>
<td>0.99</td>
</tr>
<tr>
<td>cit-patents</td>
<td>(CP) 3.77</td>
<td>16.52</td>
<td>4.38</td>
<td>0.58</td>
</tr>
<tr>
<td>Graphs500-23</td>
<td>(G5) 4.61</td>
<td>129.33</td>
<td>28.05</td>
<td>4.17</td>
</tr>
<tr>
<td>soc-LifeJournal1</td>
<td>(LJ) 4.85</td>
<td>68.99</td>
<td>14.23</td>
<td>2.27</td>
</tr>
<tr>
<td>DBPedia</td>
<td>(DP) 11.47</td>
<td>116.60</td>
<td>10.16</td>
<td>3.87</td>
</tr>
<tr>
<td>Twitter-2010</td>
<td>(TW) 41.65</td>
<td>1468.37</td>
<td>35.25</td>
<td>47.60</td>
</tr>
<tr>
<td>Friendster</td>
<td>(FR) 65.61</td>
<td>1806.07</td>
<td>27.53</td>
<td>58.94</td>
</tr>
</tbody>
</table>

Tab. 2: Graph datasets used in measurements.
Evaluation

(a) Pagerank.

(b) Union Find.

(c) Label Propagation.

(d) Strongly-Connected Components.

Fig. 4: Total runtimes.
Twitter Troll Analysis
Russian trolls went on attack during key election moments

by BEN POPKEN

https://www.nbcnews.com/tech/social-media/russian-trolls-went-attack-during-key-election-moments-n827176
Russian trolls went on attack during key election moments

by BEN POPKEN

Ben Popken
@bpopken

Huge props and thank you to @neo4j and their @mdavidallen and @lyonwj for helping compile and analyze the deleted twitter data, surfacing trends and uncovering new angles.

https://www.nbcnews.com/pages/author/ben-popken
345k **Tweets**, 41k **Users** (454 Russian **Trolls**)
Your typical American Citizen?

Cleveland Online
@OnlineCleveland
Breaking news, weather, traffic and more for Cleveland. DM us anytime. RTs not endorsements
📍 City of Cleveland, USA

@ClevelandOnline

@LeroyLovesUSA

Your typical Local News Publication?

Your typical Local Political Party?

Tennessee
@TEN_GOP
Unofficial Twitter of Tennessee Republicans. Covering breaking news, national politics, foreign policy and more. #MAGA #ZP
Your typical **Russian Troll**

@ClevelandOnline

Your typical **Russian Troll**

@LeroyLovesUSA

Your typical **Russian Troll**

@TEN_GOP
The Russia Investigations: Mueller Indicts The 'Internet Research Agency'

February 17, 2018 · 7:00 AM ET

Philip Ewing

This week in the Russia investigations: A major new indictment from the special counsel's office that charges thirteen individuals and three companies and shakes up the political rhetoric as new facts are revealed in the sprawling imbroglio.

Justice Department special counsel Robert Mueller prefers to let his work do the talking for him. On Friday, he delivered a stemwinder.

Thirteen Russians and three Russian entities were indicted by a federal grand jury in connection with the attack on the 2016 election. The indictment lays out a number of detailed allegations against the Internet Research Agency located in St. Petersburg and against individuals who owned, controlled, funded or worked for the organization.
MATCH (u:User {screen_name: "LeroyLovesUSA"})-[[:POSTED]->(t:Tweet)-[:HAS_TAG]->(ht:Hashtag {key: "thanksobama"})]
RETURN *
MATCH (t:Tweet)<-[:POSTED]-(u:Troll)
RETURN t.dayOfYear AS day, COUNT(*) AS num ORDER BY day

https://www.nbcnews.com/tech/social-media/russian-trolls-went-attack-during-key-election-moments-n827176
Hashtags

- Use of hashtags to gain visibility and insert into conversation

- @WorldOfHashtags
  - #RejectedDebateTopics

https://www.nbcnews.com/tech/social-media/russian-trolls-went-attack-during-key-election-moments-n827176
1. MATCH (tr:Troll)-[:POSTED]->(tw:Tweet) WITH tr, tw
2. OPTIONAL MATCH (tw)-[:RETWEETED]->(rt:Tweet)
3. OPTIONAL MATCH (tw)-[:IN_REPLY_TO]->(irp:Tweet)
4. RETURN distinct tr.screen_name as screen_name, count(tw) as totalTweets,
   count(rt) as totalRetweets, count(irp) as totalReplies,
   (count(tw) - (count(rt) + count(irp))) as originalContent
5. ORDER BY totalTweets DESC;
tweets vs. hour of day

Moscow business hours

tweets

hour of day
MATCH (r1:Troll)-[:POSTED]->(t1:Tweet)<-[[:RETWEETED]]-(t2:Tweet)<-[[:POSTED]]-(r2:Troll)
MATCH (r1:Troll)-[:POSTED]->(:Tweet)
  <-[:REtweetED]-(:Tweet)<-[:POSTED]-(r2:Troll)
WITH r1, r2, count(*) as freq WHERE freq > 5
RETURN r1, r2, apoc.create.vRelationship(r1, 'AMPLIFIED', {freq: freq}, r2) as rel
MATCH (r1:Troll)-[:POSTED]->(:Tweet)
  <-[:REtweeted]-(:Tweet)<-[:POSTED]-(r2:Troll)
WHERE r1 <> r2
WITH r1, r2, count(*) as freq
CREATE (r2)-[:AMPLIFIED {weight:freq}]->(r1)
Weighted In-Degree Centrality

match (t:Troll)<-[r:AMPLIFIED]-(t) with t, sum(r.weight) as total return t.screen_name, total order by total desc limit 5

<table>
<thead>
<tr>
<th>t.screen_name</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEN_GOP</td>
<td>239</td>
</tr>
<tr>
<td>NotRitaHart</td>
<td>107</td>
</tr>
<tr>
<td>GiselleEvns</td>
<td>104</td>
</tr>
<tr>
<td>tpartynews</td>
<td>98</td>
</tr>
<tr>
<td>DaileyJadon</td>
<td>84</td>
</tr>
</tbody>
</table>
PageRank on Inferred AMPLIFIED Graph

CALL algo.pageRank(
"MATCH (r:Troll) WHERE exists( (r)-[:POSTED]->() )
RETURN id(r) as id",
"MATCH (r1:Troll)-[:POSTED]->(:Tweet)
<-[:RETWEETED]->(:Tweet)<-[:POSTED]-(r2:Troll)
RETURN id(r2) as source, id(r1) as target",
{graph:'cypher'}
)
PageRank on Inferred AMPLIFIED Graph

MATCH (t:Troll) WHERE EXISTS (t.pagerank) RETURN t.screen_name, t.pagerank ORDER BY t.pagerank DESC LIMIT 5

<table>
<thead>
<tr>
<th>&quot;t.screen_name&quot;</th>
<th>&quot;t.pagerank&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;TEN_GOP&quot;</td>
<td>10.3859635</td>
</tr>
<tr>
<td>&quot;TheFoundingSon&quot;</td>
<td>8.281644000000002</td>
</tr>
<tr>
<td>&quot;GiselleEvns&quot;</td>
<td>6.4624315</td>
</tr>
<tr>
<td>&quot;tpartynews&quot;</td>
<td>6.3289815</td>
</tr>
<tr>
<td>&quot;ChrixMorgan&quot;</td>
<td>4.231436500000001</td>
</tr>
</tbody>
</table>
Graph Visualization

Based on metrics computed by graph algorithms
Graph Visualization

Centrality & community detection
AMPLIFIED relationships

Node size → PageRank
Color → community detection
Rel Thickness → weight
Graph Visualization

Graph visualizations powered by vis.js with data from Neo4j.

```javascript
var config = {
  container_id: "viz",
  server_url: "bolt://localhost:7687",
  server_user: "neo4j",
  server_password: "sorts-swims-burglaries",
  labels: {
    "Character": "name",
    "Character": {
      "caption": "name",
      "size": "pagerank",
      "community": "community"
    }
  },
  relationships: {
    "INTERACTS": {
      "thickness": "weight",
      "caption": false
    }
  },
  initial_cypher: "MATCH (n)-[r:INTERACTS]->(m) RETURN n,r,m"
};

viz = new NeoVis.default(config);
viz.render();
```

https://github.com/neo4j-contrib/neovis.js
2 days later - IRA taken to court and indicted

Feb 14:

GET THE DATA:

- Regular reader? Download streamlined spreadsheet (29 mb) with just usernames, tweet and timestamps. We recommend you right click on links and select “save link as” or similar, otherwise it may take a long time to load in your browser.

- View full data for ten influential accounts in Google Sheets

- Researcher? Download tweets.csv (50 mb) and users.csv with full underlying data

- Explore a graph database in Neo4j

Feb 16:

IN THE UNITED STATES DISTRICT COURT FOR THE DISTRICT OF COLUMBIA

UNITED STATES OF AMERICA v.

INTERNET RESEARCH AGENCY LLC A/K/A MEBASAIYEL LLC A/K/A GLAVSET LLC A/K/A MIXINFO LLC A/K/A AZMUT LLC A/K/A NOVINFO LLC, CONCORD MANAGEMENT AND CONSULTING LLC, CONCORD CATERING, YEVGENY VIKTOROVICH PRIGOZIN, MIKHAIL IVANOVIICH BYYSTROV, MIKHAIL LEONIDOVICH BURCHIK A/K/A MIKHAIL ABRAMOV, ALEKSANDRA YURYEVNA KRIVKOVA, ANNA VLADISLAVOVNA BOKACHIEVA, SERGEY PAVLOVICH POLOZOV, MARIA ANATOLYEVNA BOVDA A/K/A MARIA ANATOLYEVNA BELYAEVA, ROBERT SERGEYEVICH BOVDA, DZHEYKHAH NASIMI OGLY ASLANOV A/K/A JAYHOON ASLANOV A/K/A JAY ASLANOV, VADIM VLADIMIROVICH PODKOPAEV, GLEB KOREVICH VAISLCHENKO, IRINA VIKTOROVNA KAVORZINA, and VLADIMIR VENKOV.

CRIMINAL NO. (18 U.S.C. §§ 2, 371, 1349, 1028A)

https://www.nbcnews.com/tech/social-media/now-available-more-200-000-deleted-russian-troll-tweets-n844731
Surprising Takeaways

- Amplifying w/ retweets
- Used social media automation tools
  - Not necessarily live responses
- Meddling in elections is just another 9-5 job
- Data availability

- See lyonwj.com for code, etc.
- https://www.nbcnews.com/tech/social-media/russian-trolls-went-attack-during-key-election-moments-n827176
neo4jsandbox.com

Neo4j Browser: https://10-0-1-194-33031.neo4jsandbox.com/
Direct Neo4j HTTP: http://54.237.227.207:33031/browser/
Username: neo4j
Password: recognition-bins-procurement
IP Address: 54.237.227.207
HTTP Port: 33031
Bolt Port: 33030
Expires: 8 days, 19 hours, 42 minutes

https://hackernoon.com/six-ways-to-explore-the-russian-twitter-trolls-database-in-neo4j-6e52394c38f1
Thank You

Questions?