Neo4j Demo

0. Motivation

EpiHiper works on contact networks and output modified contact networks as well as some addditional simulation results. Can we maintain these data into data structures and make analyses on the networks and results over time? A series of questions need to be answered such as how to store the contact networks (both in memory and disc), how to make queries over the networks, how to handle the time series, can we do these efficiently, and etc.

1. Environment Preparation

- **Mandatory**:
  - Jave JDK 11 (Oracle or OpenJDK
- **Optional**:
  - Docker/Singularity (only when Neo4j is deployed on VMs)

2. Installation

https://neo4j.com/docs/operations-manual/current/installation/

3. Singularity Image Preparation

Converted from official Docker image.

- **Official Docker images**
  - Download from here: https://hub.docker.com/_/neo4j/?tab=tags
  - Mind the Docker tag you want to use. `neo4j:latest` and `neo4j:community` are fine for the free version.
  - If you want to try Docker images first, read this: https://neo4j.com/developer/docker/
- **Singularity Documents**
  - Find your Singularity version from here: https://sylabs.io/
  - Need a *container definition file*. What is it? See here: https://sylabs.io/guides/3.7/user-guide/definition_files.html

- Our definition file saved in name `neo4j_singularity.def` :

```
Bootstrap: docker
From: neo4j:latest

%post
chmod -R 777 /var/lib/neo4j
echo "dbms.connector.bolt.listen_address=0.0.0.0" >> /var/lib/neo4j/conf/neo4j.conf
echo "dbms.connector.https.listen_address=0.0.0.0" >> /var/lib/neo4j/conf/neo4j.conf
echo "dbms.connector.http.listen_address=0.0.0.0" >> /var/lib/neo4j/conf/neo4j.conf
echo "dbms.security.auth_enabled=false" >> /var/lib/neo4j/conf/neo4j.conf
echo "dbms.security.procedures.unrestricted=my.extensions.example,my.procedures.*,apoc.*" >> /var/lib/r
echo "dbms.security.procedures.allowlist=apoc.coll.*,apoc.load.*,gds.*,apoc.*" >> /var/lib/neo4j/conf/r
echo "apoc.import.file.enabled=true" >> /var/lib/neo4j/conf/neo4j.conf
echo "dbms.memory.heap.initial_size=31g" >> /var/lib/neo4j/conf/neo4j.conf
echo "dbms.memory.heap.max_size=31g" >> /var/lib/neo4j/conf/neo4j.conf
echo "dbms.memory.pagecache.size=80g" >> /var/lib/neo4j/conf/neo4j.conf

%startscript
tini -s -g -- /docker-entrypoint.sh neo4j
```

  `Bootstrap: docker` tells the builder to search for Docker images. `From: neo4j:latest` gives the exact tag of the desired Docker image. In `%post` section, some networking and plugin settings are configured. `%startscript` section gives the essential command to start the Neo4j server. `docker-entrypoint.sh` is used to check and set up some runtime configurations for Neo4j, and finally starts the server. The file can be found here: https://github.com/LeSaRDe/neo4j_test/blob/master/docker-entrypoint.sh `dbms.security.auth_enabled=false` disables the authentication of Neo4j. And the last four settings are suggested by the Neo4j team.
- The definition file can also be found here: https://github.com/LeSaRDe/neo4j_test/blob/master/neo4j_singularity.def
- Build Singularity image:

```
sudo singularity build neo4j_community.sif neo4j_singularity.def
```

  May have to run this command locally if you don't have the root on Rivanna. More about `singularity build` can be fonud here: https://sylabs.io/guides/3.7/user-guide/build_a_container.html
- Upload to Rivanna:
  Typically, it is put in here:
  `/project/biocomplexity/mf3jh/neo4j_workspace/img/neo4j_community.sif`

4. Run Neo4j SIF As a Server on Rivanna Computing Node

For convenience, the runtime folders are exposed to outside (e.g. config files and logs).

- Bash Script for Running Neo4j: https://github.com/LeSaRDe/neo4j_test/blob/master/launch_neo4j.sh
- Slurm Script for Running Neo4j on Rivanna: https://github.com/LeSaRDe/neo4j_test/blob/master/launch_neo4j.slurm

5. Access Neo4j Server From Another Machine

Typically, `cypher-shell` is used for accessing Neo4j servers. There are two steps.

- (1) Run Singularity Shell. Bash script for this: https://github.com/LeSaRDe/neo4j_test/blob/master/launch_cypher_shell.sh
- (2) Run `cypher-shell` inside the Singuilarity shell using the command:

  ```
  cypher-shell -a [NEO4j HOST NAME]
  ```

  `[NEO4j HOST NAME]` is the host name of the machine running the Neo4j server.

6. Driver for Python Interface

- All communications between Python and Neo4j need a driver for bridging (as most DB drivers do).
- TICSMTC: https://github.com/LeSaRDe/neo4j_test/blob/master/neo4j_ops.py
- More details about driver construction can be found here: https://neo4j.com/docs/api/python-driver/current/api.html#driver-construction

7. Create Graph DB, Indexes and Constraints

- The Community edition does not support creating databases (though by some trick techniques this can be done anyway).
- For the Community edition, the only database can be used is the built-in one named `neo4j` .
- The Community edition does not support creating constraints either.
- Creating indexes is supported by both editions.
- Details about database management can be found here: https://neo4j.com/docs/cypher-manual/current/databases/

- Details about constraints can be found here: https://neo4j.com/docs/cypher-manual/current/constraints/
- Details about indexes can be found here: https://neo4j.com/docs/cypher-manual/current/indexes-for-search-performance/
- Example of creating database:

```
create database DBNAME if not exists
```

- Example of creating constraint:
  The uniqueness of PID for each person (as a node in DB).

```
create constraint pid_unique if not exists on (n:PERSON) assert n.pid is unique
```

- Example of creating index:
  Create a BTree index for the *age* attribute of person.

```
create btree index idx_age if not exists for (n:PERSON) on (n.age)
```

- Take a look at the data structures and indexes for EpiHiper contact networks. TICSMTC: https://github.com/LeSaRDe/neo4j_test/blob/master/neo4j_ops.py

8. Create Nodes

Two different methods were tried.

- (A) Use `CREATE` with/without multithreading. Commit batch by batch.
- (B) Use `apoc.periodic.iterate()` and `apoc.load.csv()` with parallelization.
- Experimental Settings:
  - WY data: /project/bii_nssac/COVID-19_USA_EpiHiper/rivanna/20211020-network_query/
  - Data description: https://github.com/NSSAC/EpiHiper-network_analytics
  - Number of nodes: $548,603$
  - Number of edges in the initial contact network: $21,685,569$
  - 32 Threads (Rivanna computing node, exclusive mode).
  - Batch size: $100,000$
- Some Results:
  - With/without predefined indexes, running times are similar.
  - Method (A): < 1 min (no significant difference between `cypher-shell` and Python), $30 \sim 60$ seconds for each thread.
  - Method (A) in a single thread: > 70 seconds.
  - Method (B): several seconds.

    - Creating indexes after creating nodes is very fast in this case, though the running time may vary for larger data.
- `apoc` , though also uses parallelization, processes data in some different ways.
- TICSMTC: https://github.com/LeSaRDe/neo4j_test/blob/master/neo4j_ops.py

9. Create Edges for Initial Contact Network

Similar to creating nodes.

- Some Results:
    - Method (B): ~ 15 minutes.
    - Method (A): When loading in VA data, it took more than 30 hours using a single thread.
    - Method (B): For the same VA data, it took about 4.5 hours.

10. Create Edges for Intermediate Contact Networks

Similar as before.

- A key concept capturing the time series is the `occur` attribute at edges. `occur = -1` for the initial contact network, and `occur > 0` for intermediate networks.
- Some Results:
    - Method (B): ~ 1.7 hours for loading in 5 intermediate contact networks in sequence.

CN Edge File	Lines	Time
Initial	21685569	15 min
network[0]	21685569	23 min
network[1]	21685569	18 min
network[2]	20111489	17 min
network[3]	20111489	17 min
network[4]	20111489	30 min

    - The input data to Neo4j (initial contact networks + 5 intermediate contact networks) is about 7 GB. The database size now is about 62GB. The expansion ratio is about 9.

11. Create SQLite DB for EpiHiper Output

When using Neo4j to process EpiHiper's data, my methodology is *keep networks inside Neo4j and the rest outside*. Sounds trivial, but it may become much less trivial when things turn complex.

- Create DB, Load output data, Create indexes.
- Running time is trivail so far.
- Take a look at the data structure and indexes. TICSMTC: https://github.com/LeSaRDe/neo4j_test/blob/master/neo4j_ops.py

12. A Concrete Scenario

Pull out the distribution of durations w.r.t. an exit state.

- Given an exit state: `Isymp_s`
- Fetch PIDs over time for this exit state (SQLite).
- Find contacts incoming to these PIDS for each time point. Translated: query incoming 1-nearest-neighbors of a given set of nodes for each time point. (Neo4j)
- Get the distribution of durations.
- TICSMTC: https://github.com/LeSaRDe/neo4j_test/blob/master/neo4j_ops.py

13. Link with SNAP/CINES

Execute a query on Neo4j, retrieve the subgraphs, convert subgraphs to SNAP TTables batch by batch. This is useful for some graph algorithms. Note that constructing TTables can be tricky, and the documents of SNAP is not perfect.

- More about TTables:
  - https://snap.stanford.edu/snappy/doc/reference/conv.html
  - https://snap.stanford.edu/snappy/doc/reference/table.html
- TICSMTC: https://github.com/LeSaRDe/neo4j_test/blob/master/neo4j_ops.py

14. Another Example (fast)

- Query population distribution grouped BY `age_group`
- Cypher code:

```
match (n)
with distinct n.age_group as age_group
unwind age_group as each_age_group
match (m)
where m.age_group = each_age_group
return each_age_group, count(m)
```

- Results:

each_age_group	count(m)
"a"	213047
"p"	35874
"s"	97906
"g"	82645
"o"	119131

Running time: 2498 ms

15. One More Example (use apoc graph properties)

Query incoming degrees of infected people at a given time point.

- Get infected PIDs
- Cypher code:

```
unwind $infect_pid as infect_pid
match (n:PERSON {pid: infect_pid})
return infect_pid, apoc.node.degree(n, "<CONTACT")
```

- Run it out. TICSMTC: https://github.com/LeSaRDe/neo4j_test/blob/master/neo4j_ops.py

16. The Last Example (Slow)

- Query source activity distribution
- Cypher code:

```
match ()-[r]->()
with distinct r.src_act as src_act
unwind src_act as each_src_act
match ()-[q]->()
where q.src_act = each_src_act
return each_src_act, count(q)
```

- Results:

each_src_act	count(q)
"1:1"	22796760
"1:5"	15745284
"1:3"	4761003
"1:2"	38399970
"1:4"	39981876
"1:7"	157923
"1:6"	3548364

Running time: 1154917ms = ~19min