**Dgraph benchmark**

**1. Environment setup**

1) Install Dgraph

curl https://get.dgraph.io -sSf | bash

2) Run Zero server (cluster leader):

dgraph zero

3) Run Alpha instance (one replica):

dgraph alpha --lru\_mb 16000 --zero localhost:5080

lru is an operational cache

4) Clone the repository with the test script

git clone <https://github.com/Kubera2017/dgraph-benchmark.git>

5) Run the script

go run dgraph\_v2.go

**2. Script and configuration**

The script simulates a crawl process and do the following.

1) Open connection to the database

2) Clean up all previous data and schema

3) Apply the schema

4) In loop from 0 to “batchCount”:

- batch size depends on batchSize

- generates randomly domainFrom, uriFrom, domainTo, uriTo, anchorText, rel

- format the transaction and send it to the database

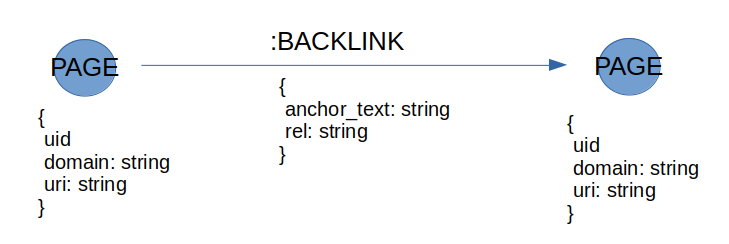
- if uriFrom is not exist – create it

- if uriTo is not exist – create it

- create backlink between uriFrom and uriTo

- optionally breaks the loop if “breakAt” is set more than zero

**3. Graph schema**

uri: string @index(hash) .

domain: string @index(exact) .

**4. Results**

**- On average, how long does a batch insert of 10K new records take?**

On average - 50 seconds. But there is strange negative correlation between batch size and insertion time, take a look at **batch\_size.xlsx** in the repository.

1K records insertion – CPU utilization 100-150%, RAM usage – 2 GB

10K records insertion – CPU utilization 130-200%, RAM usage – 5 GB

Dgraph issues discussion on GitHub says that we need launch several replicas on one machine with many cores and send batches to these replicas simultaneously to get all cores loaded.

**- What's the total size of the database on disk?**

DB size forecast is 90 GB based on results of insertion 1M - 4M records. See **db\_size\_measurements.xlsx** in the repository.

Current time to build 1B graph is minimum 144 hours (using one replica on one multi-core machine).

**4. Additional observations**

If we remove index on domain it reduces batch insertion time by ~20%

If we remove backlink edge properties (anchor\_text, rel) it reduces batch insertion time by ~5%