Neo4j and MySql comparison experiment

With caching

The experiment is run on a scientific linux machine with both databases running on it locally. The Neo4j server is embedded in the java application and the MySql server is running on it, too.

The experiment is performed separately for neo4j and mysql, to exclude any possibility of interaction between them. Before the experiment is performed and any measurements are recorded the system and the database is warmed up. The warm up consists of connecting to an instance of mysql or an instance of neo4j graph and running some queries on them. The queries are taken in order from the actual queries that will be executed to allow the database to do some caching, and the length of the warm up is at least ten seconds.

The experiment is performed on random trees that have certain size and depth. The trees are generated using random seeds, so the same tree can be generated again using the same depth, size (number of vertices) and random seed. It is planned to use trees of sizes 1 000, 10 000, 100 000, 1 000 000 and depths 30, 100, 3 000, 10 000, respectively. For each size/depth combination there are 10 different trees generated from different random seeds. The number of trees could be increased later on, if the experiment execution times allows for it.

Two types of queries are run for each tree: finding neighbors and finding lowest common ancestors. The find neighbors query takes a vertex and finds all its neighbors exactly at given depth. The find lowest common ancestors query takes two vertices and finds their lowest common ancestors or terminates upon reaching a given depth. Both queries use depths 5, 10, 15, 20, 25.

Due to some queries being very short they are executed and measured together in sets. Only the whole execution time of the set queries and nothing else is captured. The aim is to have the execution time of the set to be at least 20 ms, preferably around 100. Of course the same sets are used when measuring mysql and neo4j performance. A set still might contain only one query if it is long enough for both mysql and neo4j.

Each set contains queries only of the same type and depth, so an average execution time for the query type and depth is measured on each tree. Multiple different sets of the same depth/type will be used if the time allows for it to see how the results vary.

The whole experiment will be repeated enough times to determine how consistent the results are.

Summary

1. Start java via shell script and open a graph (tree) in mysql/neo4j.
2. Warm up.
3. Execute all query sets of that graph and record their times together with the parameters of all queries.
4. Quit and repeat step 1 with other graphs.
5. Repeat the whole experiment.

Without caching

Another similar comparison experiment should be performed, except that the database shouldn't be warmed up and only a single query is executed and measured in one java instance.

Data analysis

- Calculate the mean and confidence intervals of 95% for queries of the same type and depth on trees of the same size and depth.
- Make separate scatter plots for the query time against tree size and against the tree depth, and against the depth of the query for both databases. Try curve fitting to see if there is a correlation between them.
- Calculate the time differences (mysql time divided by neo4j time) to see how many times one outperforms the other. These differences are then plotted against the tree size to see how well they scale.

Running the experiment

The folder Results/Scripts, contains the shell scripts used in running the experiments:

- rte_1 – launches all experiment runs for the trees of size 100.
- rte_2 – launches all experiment runs for the trees of size 1000.
- rte_3 – launches all experiment runs for the trees of size 10000.
- rte_4 – launches all experiment runs for the trees of size 100000.
- rte_5 – launches all experiment runs for the trees of size 1000000.

The structure of these files is as follows, the seeds argument specifies what random seed to for building the tree and it runs the main method of the RandomTreeExperiment class. The second argument specifies weather to rewrite the tree or to check if this tree has been already built (rw tells to rewrite, anything else not to). The third argument specifies the implementation to use, so it is either “neo4j” or “mysql”. The fourth all and all following arguments specify the seed(s). The first argument specifies the file that describes the experiment.

The files describing the experiments are in the Results folder:

- rte_100_10_1
- rte_1000_30_1
- rte_10000_100_1
- rte_100000_300_1
- rte_1000000_1000_1

They are formatted as follows. The first line is the size of the tree and the second is the depth. The third line specifies the number of vertices N_i to index for querying purposes. The vertices are indexed, so they could be retrieved for passing as parameters to queries, but usually there is no need to specify these as all vertices used in queries are indexed automatically, so usually it is 0. It is useful in case it is

planned to add additional queries on other vertices . The next line (skipped if N_i is 0) contains the numbers of vertices to index. The numbers are given to vertices in the order they are added to the tree, so root is 0 and the last added node is size – 1.

The next line specifies the count M of query sets to execute. Each of the next M lines describes one query set. These lines start with the query type “FND” or “LCA” for find neighbors at depth and lowest common ancestors query types, respectively. The next integer is the depth depth parameter of these queries and then the next integer is the count N of queries. The following N (or 2N numbers for LCA) are the numbers of the vertices to use in each query.

The results of the experiment are written appended to this file.

The neo4j databases are created/opened by the Neo4jFactory class method createGraph, it creates the database in the user_home/graphs directory, where user is the user_home directory of the user.

The mysql databases are created/opened by the MySqlFactory class method createGraph. The constructor of the MySqlFactory needs 3 arguments – the link to the mysql database, users name and password. The factory is created in the main method of the RandomTreeExperiment, where these three arguments can be changed.

Processing the results

All the results where concatenated in one file all_results and processed using statistics package R. The commands used to perform the calculations, obtain plots and confidence intervals are in the file Results/R_for_RTE. The plots where exported to pdf and merged into one document GraphDBPlots.pdf.