Dynamic Noise and Pollution Campus Map

/progress report/

Work Done Previous Week

- Parsing + loading models from JSON (custom deserialisation)
- MySQL tests (benchmarks: 5000, 60000, 200000 el. set)
    - Results: fast update, retrieve and delete, slower insertion; larger data sets lead to a significant slow-down; however, when caching is used, very fast execution
- MongoDB tests (benchmarks: 5000, 60000, 200000, 5000000 el. set)
    - Results: extremely fast insertion -> 200 000 records for around 20 sec.; however, caching is significantly less efficient while MySQL guarantees better performance in the long run (in my opinion, looking at the test results)

Work Under Way

- visualise data (simulation)

Known Issues + Resolutions

- Auto-wiring the MongoDB template to the controller -> resolved by using the @Autowired Spring annotation for bean parameters.
- Configuring the models to work with MongoDB, especially the Timestamp object (tried to use mongo's BSONTimestamp) -> resolved by reverting back to java.sql.Timestamp

Test Conclusions:

Mongo DB does an excellent job regarding data insertions - its capabilities/ insertion algorithm/ have no match considering this. Data retrieval, update and delete of single records is also very fast, however, it is difficult to argue if much faster than *MySQL* .

All of the benchmarks have been completed without caching mechanisms being used. Not surprisingly, *Mongo DB* has shown around x1.5 up to x2 faster data manipulation than MySQL and this tendency is present for all of the data sets (I did not even think of running a 5 million data set insertion script for the SQL DBMS).

Efficient data caching is one of the main advantantages of *MySQL* in this comparison. Although I have not uploaded results with memory caching in use, I noticed that query execution (especially database reading) for more lengthy operations dramatically decreases up to a multiplier of 2 after each consecutive call to the database, which will inevitably lead to a very improved efficiency in a long run. I would assume that it would be even much better than mongo. On the other hand, *Mongo DB* delivers a great first impression for the users who will visit the page more rarely (in other words, not make use of in-memory cache).

There is no need to consider the relational db structural benefits against the document-based, as the plans for MySQL did not involve querying more than a table at a time (or 2) and this would not affect the final decision much.

I did a quick search for similar tests/evaluations, and found some information confirming my observations so far:

- http://www.moredevs.ro/mysql-vs-mongodb-performance-benchmark/

Repository Benchmarks:

- DB System Choice: https://github.com/ppyordanov/Dynamic-Noise-and-Pollution-Map/issues/27
- NoSQL DBMS Evaluation: https://github.com/ppyordanov/Dynamic-Noise-and-Pollution-Map/issues/30
- MySQL Evaluation: https://github.com/ppyordanov/Dynamic-Noise-and-Pollution-Map/issues/29

Petar Yordanov, 04.11.2014