Experiment's description:

In the java application a scenario is simulated, in which random input user data is generated and buffered, and then bulk-loaded into DB. In the meanwhile, there is a user agent that requests some random data. PostgreSQL is used for database. Application uses only one database, containing only one table (Database index on ID column (integer value-Primary key)).

The Java application consists of following threads:

-CreateTextFileThread:

In this thread, a text file is created that contains 10,000 rows of sample user data. Each row has 10 columns, some varchar (constant value in all rows), some numeric (random value). This is used as a simulation for having incoming user input data and buffering it into a file. This way, we can later use "COPY" operation for bulk-loading which is faster than series of "INSERT" operations for all rows. (See chapter 14. Performance Tips in PostgreSQL's documentation for more information.)

-InsertRowsThread:

Here, the created text file is copied into database. As in PostgreSQL's documentation was mentioned that creating index on pre-existing data is faster than updating index incrementally as each row is loaded, in this experiment, Before loading the data, we first dropped indexes to see how it effects entire loading duration. So, in general this thread does the following steps:

1. removes the indexes
2. loads the entire text file (the buffer) into table using "COPY" command
3. adds the indexes again
4. runs "ANALYZE" command

Running "ANALYZE" after bulk loading, ensures that the planner has up-to-date statistics about the table, which results in better decision making during query planning. (See chapter 14. Performance Tips in PostgreSQL's documentation for more information.)

-The RequestRow Threads:

These threads just send random queries based on table's numeric columns and then go to sleep for 1 or 2 seconds after getting the results of the query.

The CreateTextFile Thread and InsertRows Thread take turns, so that data is loaded after new text file is created. Queries are sent simultaneously. Each time a thread operates and goes to sleep is called a cycle in excel files.

Other classes are there just to run threads or to save the results.

The excel files show the results of running the program for about 2 hours.

1.RequestIDsleep1: users ask for some rows (random select) and get results (duration and number of results is shown , cycle count : how many time the select operation is performed)

2.TextBufferExcel: shows how long does it take (in milliseconds) to create the text buffer file (10,000 rows to be copied into DB)

3.CopytoDBExcel: shows how long does it take to copy the buffer to DB (drop index , copy to table, add index again, analyze)

In all excel files duration is reported in millisecond.

In first file which is RequestIDsleep1, Query duration and number of results are growing because the numeric values in condition for random select are growing. So, in this experiment we can not actually conclude that whether time to process and answer a query is growing as the database grows or not.