Assignment 6

Concurrency Tuning

Database Tuning

Due date: June 5, 2020
Grading: 5 points

Notes

- It is suggested that you also have a look at the report template before you start working on the assignment.

Access Parameters for PostgreSQL

- Host: `biber.cosy.sbg.ac.at`
- Port: `5432`
- Database: `dbtuning_ss2020`
- User/Password: you should have received them via email

The database server (`biber.cosy.sbg.ac.at`) is accessible only from within the university network. If you would like to work from home, please connect to `fanny.cosy.sbg.ac.at` via `ssh`. Java, the PostgreSQL client, and Python are installed on this machine.

Support

If there are any ambiguities or problems of understanding regarding the assignment, you have the following possibilities to clarify them:

- Upon request via email (martin.schaeler@kit.edu)

In this assignment you will explore the concurrency control features of PostgreSQL.

A company with 100 employees pays the salaries at the end of the month. The account of the company (account number 0, initial balance 100) and the accounts of all employees (account numbers 1 to 100, initial balance 0) are with the same bank. The payment transactions should run concurrently. Here are two approaches to solve the problem:

(a) For each employee $1 \leq i \leq 100$ run the following transaction:
$e \leftarrow$`SELECT balance FROM Accounts WHERE account=i`
`UPDATE Accounts SET balance=`$e + 1$ `WHERE account=i`
$c \leftarrow$`SELECT balance FROM Accounts WHERE account=0`
`UPDATE Accounts SET balance=`$c - 1$ `WHERE account=0`

(b) For each employee $1 \leq i \leq 100$ run the following transaction:

```
UPDATE Accounts SET balance=balance+1 WHERE account=i
UPDATE Accounts SET balance=balance-1 WHERE account=0
```

Solution (a)

Run solution (a) with isolation level `READ COMMITTED`. Compare throughput and correctness for different numbers of concurrent transactions, ranging from 1 to 5. The correctness is defined as $(c_1 - c_2)/100$, where c_1 and c_2 are the balances of account 0 before and after running all transactions, respectively. Repeat the experiment with isolation level `SERIALIZABLE`.

Note: If a query is rolled back, restart it until it commits. Java and Python template code is provided in `account.zip`.

Solution (b)

Run solution (b) with isolation level `READ COMMITTED`. Compare throughput and correctness for different numbers of concurrent transactions, ranging from 1 to 5. The correctness is defined as $(c_1 - c_2)/100$, where c_1 and c_2 are the balances of account 0 before and after running all transactions, respectively. Repeat the experiment with isolation level `SERIALIZABLE`.

Discussion

Discuss the outcome and explain the difference between the isolation levels in PostgreSQL with respect to your experiment. The following information sources might be useful:

- Lecture notes:
  `https://dbresearch.uni-salzburg.at/teaching/2018ss/dbt/dbt_04-handout-1x1.pdf`
- PostgreSQL documentation:
  `https://www.postgresql.org/docs/9.6/static/transaction-iso.html`

Report

1./2. Give throughput and correctness for both solutions, (a)/(b), with serialization level `READ COMMITTED` and `SERIALIZABLE`, respectively.

3. Discuss the outcomes and explain the difference between the isolation levels in PostgreSQL with respect to your experiment. In own words: Explain how PostgreSQL deals with updates in different isolation levels, within a transaction and within a single SQL command.

Please indicate the average time per group member that was spent solving this assignment. The time that you indicate will have *no* impact on your grade.

Grading scheme:

Category	**Max. Points**
Description of your setup	0.5
Execution of experiments (Solution (a) and (b))	1.5
Discussion of results	1.5
Isolation level discussion	1.5

Reminder: Additional questions about the involved topics/techniques will be asked during the meeting.

Important: If the grading scheme is unclear, ask the instructor!