CSci 5708

Lab 2

Fall 2016

Due Thursday, 11/03/2016 13:00

(Deadline for PostgreSQL installation: 10/25/2016)

Note

1. Please read this document carefully for the things to do and submission guidelines.
2. Not following the submission guidelines will result in penalty.
3. This lab must be done in **groups of two**. Use moodle
4. The best way to get your questions answered is to ask them on **moodle** forums. That way it helps other students who are facing similar problems.
5. Also, make sure you follow moodle for announcements and forum question-answers.

Goal

In this lab, you will experience modifying PostgreSQL, a major open-source DBMS. Your goal is to implement a different buffer replacement policy. The current version of PostgreSQL uses a **clock sweep algorithm with LRU** to determine which buffer page should be replaced. In this lab, you are asked to implement a new buffer replacement policy, LIFO (Last In First Out).

For example, consider a memory with 3 buffers as follow:

Initially all 3 buffer spaces are available.

Buffer ID	🟩	🟩	🟩
Timestamp			

As the systems starts executing queries, data pages are brought into buffers. Here, timestamp is a logical entity (increasing integer). It does not have be a system time value.

Buffer ID	🟥	🟩	🟩
Timestamp	1		

Buffer ID	🟥	🟥	🟩
Timestamp	1	2	

Buffer ID	🟥	🟥	🟥
Timestamp	1	2	3

Suppose we need to bring in 4th page in memory. In such a case, where all three buffers are unavailable (and say **none of them are in use)**, we need to use buffer replacement strategy- LIFO to make space for the new page. LIFO will go through the buffer timestamps and select the buffer with "most recent" timestamp that is not being used. In this case, it will be the right-most buffer. Therefore, the memory will look like this after replacement:

Buffer ID	🟥	🟥	🟥
Timestamp	1	2	4

For this lab, most of your time will be spent in code browsing and understanding the existing source code, so make sure you get an **early start**. Reading and understanding an open-source framework is the main goal of the lab. This skill is very important once you go to industry where most likely you will spend your time modifying an existing system/framework rather than creating a new framework from scratch.

The lab must be done in a group of 2 (use moodle if you do not have a lab partner!). Please download the “data_set.zip” from the course website for the dataset of the database as well as the template of “solution.txt”. (See Submission Guidelines below).

Best Practice

Do not remove the old code of PostgreSQL. The best practice is to switch from the old code to your modification by using a debug flag and an if statement. This will help you not to reinstall the whole PostgreSQL for the subsequent labs.

Help

Since reading and understanding an open-source framework is the main goal of the lab, the TA is only allowed to do the following:

1. Help you with the installation of PostgreSQL.
2. Help you in understanding the concept of **buffer management policy** (not specific to PostgreSQL query optimizer).
3. Help you in understanding what the assignment wants you to do.

Unfortunately, the TA **is not allowed** to help you in navigating the source code (including telling you the functionality of a function or variables) since one of the goal of the lab is to make sure that you understand the source code of the query optimizer in PostgreSQL. Once you understand the source code and know where to make the change, it is very easy to solve the lab.

Installation

Download version 9.2.1 of PostgreSQL from http://www.postgresql.org/ftp/source/v9.2.1/ and install it to your Linux CSELabs or CS machines. Detailed installation instructions can be found at http://www.postgresql.org/docs/9.2/static/install-procedure.html. PostgreSQL is a large program, so you will want **at least** about 200 MB **of free space** on the machine. Sorry :(

As an additional note, **the standard "./configure" command should not be used during installation**, as students do not have permission to install anything to the user directory. You will want to specify another folder within your home directory to install PostgreSQL to, with the command:

```
"./configure --prefix=<install_path>"
```

The prefix option requires an absolute path, not a relative one. One example would be:

```
"./configure --prefix=$HOME/PostgreSQL/installation"
```

Setup

Once the program is fully installed, we need to initialize and create a database to use. To initialize databases to a certain folder, use the following command:

```
"<install_path>/bin/initdb -D <install_path>/data"
```

We can then start the PostgreSQL backend by using the following command:

```
"<install_path>/bin/postgres -D <install_path>/data"
```

We can then use this terminal to get information on what is happening in our DBMS. Open **another** terminal and use the following command to create a database to use:

```
"<install_path>/bin/createdb -h localhost <database_name>"
```

We interact with our DBMS system using a program called psql. To start psql, use this command:

```
"<install_path>/bin/psql -h localhost <database_name>"
```

You can now use this terminal to enter SQL commands and manage your database. After this, whenever you start up PostgreSQL, you should only need to run the postgres and psql commands, even after rebuilding and reinstalling.

Modification

All the operations related to buffer manager are location in src/backend/storage/buffer directory. From there, you will need to find the location of the buffer replacement policy and add your changes there. It's a good idea to get an early start in deciphering just how the buffer replacement function works. Leaving this lab for the last minute is an extremely bad idea.

Whenever you make any changes, you will need to run `"make"` and `"make install"`. Then you can run the postgres and psql command to run the database.

Printing

For verifying the correctness of your implementation, you need to print out the timestamp of all candidate (not in use) buffers as: **"Candidate buffers: <a comma separated list of timestamps>"** and also explicitly print out the timestamp of the buffer that is going to be used/replaced as: **“Replaced buffer: <timestamp>”**. Initially, all buffers will have zero timestamp.

Since the test case that is given to you is very small compared to the default memory size of PostgreSQL, it is a good idea to change the memory size in the configuration file of the PostgreSQL. The configuration is located in:
“<install_path>/data/postgresql.conf”
You will update line number 113 and make the shared_buffers = 128kB.
To test your change, run the data_set/buffer_add.sql file and look at the replaced buffer’s timestamp compared to all other candidate buffers’ timestamp. In order to run buffer_add.sql, you need to make one modification to the file. Please make sure you change the path of data_set/values10k.dat file according to its location.

The PostgreSQL online source code documentation will be an invaluable resource for this lab and the next. You can find it at http://doxygen.postgresql.org

Submission Guidelines

- **One submission per group.**
- The lab **must** be submitted in ".zip" format via Moodle with the following **naming convention**: "lab2.zip"
- There is one folder and one file that needs to be submitted (again please follow the naming convention) inside the lab2.zip:
    1. "changes": A folder that contains all files that you have changed in order to fulfill the lab. Please only submit the file itself (don't include the directory). For example: if you change a file in src/backend/foo.c, only put foo.c in the folder.
    2. "solution.txt": Modify the solution.txt template from the zip file as per your changes.

Grading

The grading will be done in the following way:

- 10 Points: Submit the assignment in the correct naming convention.
- 60 Points: Successfully make the buffer manager to use the "LIFO" policy including incorporating the printing instructions.
- 30 Points: solution.txt contains:
    - 10 Points: a brief summary of your change.
    - 10 Points: the solution contains all paths of the modified file in "changes" folder. Example in the "solution.txt" template.
    - 10 Points: a detailed description on the modification on each file. Example in the "solution.txt" template.