Exercise 2

Released: April 16, 2018 · Discussion: April 23, 2018

1 Buffer Manager — Replacement Strategies

The lecture discussed the functioning of the *buffer manager*.

- What are the responsibilities of the buffer manager?
- Describe the strategies used to fulfill these responsibilities.
- What are their advantages and disadvantages?

2 Replacement Strategies — Implementation

Write a program, in a programming language of your choice, that implements the replacement strategy of a rudimentary buffer manager. The program shall read a sequence of `pin()` and `unpin()` calls from a given file and simulate it on a buffer manager with n buffers, where n is a user supplied parameter. For each `pin()` call your program shall decide if a certain page has to be loaded from disk and if so which frame is going to be replaced. Your buffer manager shall implement at least two different strategies.

- To test your program, we provide the file `buffer_manager.trace` which contains a set of buffer manager calls of a transactional workload[1]. The attributes `pin/unpin` and `reln` describe the respective action and page number.
- Vary the number of buffers n and evaluate the efficiency of the strategies implemented by using the hit rate. Here the hit rate is the quotient of the number of requests of a buffered page to the number of requests that required to load a page from disk.

[1] The trace was created by the TPC-C benchmark on a PostgreSQL instance.

Hints:

- For a simple strategy it is sufficient to only take the `pin()` calls into account.
- The following `C++` code snippet can be used to read the trace:

```
#include <fstream>
int main () {
  std::ifstream file;
  file.open ("buffer_manager.trace");
  std::string action, dump; int page_no;
  while(file) {
    file >> action; file >> dump; file >> dump; file >> page_no;
    // action und page_no beinhalten hier die relevanten Werte
    getline(file, dump);
  }
  file.close();
  return 0;
}
```