UNIVERSITY OF WISCONSIN–MADISON
COMPUTER SCIENCES DEPARTMENT
CS 564: DATABASE MANAGEMENT SYSTEMS

Assignment 6: Buffer Manager Viewer
Due: April 29, 1996, 5 p.m.

Instructor: Raghu Ramakrishnan

1 Introduction

In this assignment, you will carry out a number of exercises involving different patterns of page accesses, using the Minibase buffer manager visualization tool. *You will carry out this assignment in teams of two with the same partner as for the first assignment.*

You will not have to write any code. This tool (view_bufmgr) reads a trace file that lists the actions of the Buffer Manager during a run of Minibase, and shows you the internal state of the buffer pool during the run, plus some statistics about the number of dirty pages, hit ratio, etc.

Each exercise corresponds to a test case that is generated by the *bmtrace* program. You can vary the number of frames available to the Buffer Manager, and the page replacement strategy (LRU, MRU, or Clock) used by the Buffer Manager during each test case. The general approach for each of these exercises is as follows:

1. Run *bmtrace* with the appropriate test number and values for other parameters and write its output to a file.

2. Run view_bufmgr to see how the buffer pool was utilized during the run, and then answer the questions. (You might have to generate several traces for a given exercise.)

However, for Question 11, you will use *sorttrace* instead of *bmtrace* to generate trace files. The use of *sorttrace* is explained in Question 11.

2 Available Documentation

The view_bufmgr tool has an on-line help feature, and additional information is included in this handout as part of exercise descriptions. The *bmtrace* program is simple to use, typing *"bmtrace -h"* at the unix prompt will tell you all you need to know to use it.

3 The Exercises

Unless otherwise stated, you are to use the default values for both the number of buffer frames (64 pages) and buffer replacement policy (LRU policy). Note that the number of page I/Os refers to number of pages read from disk and does not include the number of page writes.

1. Test Case 1: Linear scan of a single file.

   (a) What does the Buffer Manager do for each page?

   (b) What is the maximum pin count for each page?

(c) What happens when the number of frames in the buffer is smaller than the number of pages in the file? Generate another trace with 16 frames in the buffer pool (i.e., bmtrace -n 16 1), and compare with your first trace.

2. Test Case 2: Linear scan with occasional writes, using 16 buffer frames.

   (a) What happens when a dirty frame (i.e., a frame that has been modified) is replaced?

   (b) How many disk writes are done versus the number of dirty (i.e. modified) frames?

3. Test Case 3: Several linear scans of a single file. The size of the file is 24 pages.

   (a) What is the number of page I/Os when the number of frames is equal to 18 under the LRU replacement policy? What is the smallest number of additional buffer frames you need to reduce this number of page I/Os? (i.e., find the smallest $B > 18$ such that when the number of frames is equal to B, the number of page I/Os is less than your answer to the first part.)

   (b) Repeat (a) using the MRU replacement policy.

4. Test Case 4: Several scans of one file with occasional reads from another file.

   (a) What is the number of page I/Os when the number of buffer frames is equal to 22 under the LRU replacement policy?

   (b) What is the minimum number of buffer frames required so that that there are no page replacements under the LRU replacement policy?

   (c) What is the number of page I/Os when the number of buffer frames is equal to 22 under the MRU replacement policy?

   (d) What is the minimum number of buffer frames required so that that there are no page replacements under the MRU replacement policy?

5. Test Case 5: Page oriented simple nested loops scan. The sizes of the outer and inner relations are 10 and 20 pages, respectively.

   (a) What is the minimum number of buffer frames required to avoid page replacement under the LRU replacement policy?

   (b) What is the number of page I/Os incurred when the number of buffer frames is equal to 16 under each of the following replacement policy: LRU, MRU, CLK?

   (c) Repeat (b) with the number of buffer frames increased to 17. What can you say about the reductions in the number of page I/Os for each of the three replacement policies?

   (d) Repeat (b) with the number of buffer frames increased to 18. Compare the reductions in the number of page I/Os under each of the three replacement policies with your answers to (c).

6. Test Case 6: Block nested loops join, Strategy 1: Here we simulate a block nested loops join with the outer file having 40 pages and the inner file having 15 pages. The strategy is to pin as much of the first file at once as possible, then walk through the second file one page at a time.

   (a) How many page I/Os happen when the whole first file fits in the buffer?

(b) How many page I/Os happen when the whole first file does not fit in the buffer (e.g., 36 frames)?

(c) How does replacement policy affect this join strategy?

7. Test Case 7: Block nested loops join, Strategy 2: This is a variant on Test Case 6 where we read both files in blocks.

   (a) How many page I/Os happen when both files fit in the buffer (e.g., 64 frames)?

   (b) How many page I/Os happen when the second file fits but the first file does not (e.g., 36 frames)?

   (c) How many page I/Os happen when neither whole file fits (e.g., 25 frames)?

8. Test Case 8: Clustered index scan. Here we simulate fetching tuples from a file by walking its clustered index. For each index entry, the corresponding tuple in the file is read.

   (a) How many page I/Os to walk the file using the index if the whole file fits?

   (b) What is the number of page I/Os when the number of buffer frames is equal to 20? What is the reduction in the number of page I/Os when the number of buffer frames is increased to 22?

9. Test Case 9: Unclustered index scan.

   (a) How many page I/Os to walk the file using the index if the whole file fits?

   (b) What is the number of page I/Os when the number of buffer frames is equal to 20? What is the reduction in the number of page I/Os when the number of buffer frames is increased to 22?

10. Test Case 10: Index nested loops join. Here we simulate doing a file scan of a 10-page relation, and for each tuple using an index to access related tuples from a second 40-page relation. You will need to examine the source code (function indexNestedLoops in file bmtrace.C) to answer some of these questions.

    (a) What effect on the number of page I/Os do you expect when the outer relation is sorted on the join columns versus when the case when it is not?

    (b) Examine the function *indexNestedLoops* in bmtrace.C, is the index (in the simulation) clustered or unclustered?

    (c) Assuming that the join being simulated is an equijoin, what would have to be changed to accurately simulate an index on a key of Relation 2? (Note that you just need to submit a modified version of the function *indexNestedLoops* with comments indicating the changes made, and NOT the entire bmtrace.C file).

    (d) If the join was an equijoin and the outer relation was sorted on the join columns, what effect would a clustered vs. unclustered index have on the pattern of page access?

11. External Sorting. Here we simulate doing external sorting of a file. For this question, you have to use *sorttrace* to generate trace files, instead of *bmtrace*. In addition to the two parameters (number of buffer frames and replacement policy), there are three new parameters to control the trace output:

    - File Size. This refers to the number of pages in the file to be sorted. The default value is 30 pages.
    - Run size. This refers to the number of buffer frames used to sort the records; i.e., it is the size of the initial sorted runs. The default value is one page.
    - Block size. This refers to the number of buffer frames allocated for each sorted run during the merging phase. The same number of buffer frames is also allocated to hold the merged records. By varying this parameter, we can simulate block I/O for the merging phase. This parameter also influences the fanout of the merging. The default value is one page.

    To see the options available for *sorttrace*, do a "sorttrace -h".

    (a) Use the following parameter values: LRU replacement policy, number of buffer frames = 18 pages, file size = 256 pages, and block size = 5 pages. These parameters will give a merge fanout of 2. What is the total number of disk I/Os (including reads and writes) for each of the following run sizes: 1, 2, and 3 pages?

    (b) Use the following parameter values: LRU replacement policy, number of buffer frames = 18 pages, file size = 256 pages, and run size = 1 page. What is the total number of disk I/Os (including reads and writes) for each of the following block sizes: 3, 4, and 5 pages? Note that these block sizes will give a merge fanout of 2, 3, and 4, respectively.

4 Where to Find Code, etc.

Copy all the files from ~cs564-1/spring96/assign6/src into your own local directory. The files are:

- bmtrace: Program to generate trace files for Questions 1 to 10.
- sorttrace: Program to generate trace files for Question 11.
- view_bumgr: Program to view trace files.
- Makefile: Makefile to compile bmtrace.C.
- bmtrace.C: Source code to generate trace files for Questions 1 to 10.

5 What to Turn In, and When

You should turn in:

1. Brief answers to the questions in a separate file. Use any word processing software that you want, but please make the file easily readable. (Plain text is preferable.)

The assignment is due at 5PM on April 29. The solution will be made public then, and solutions turned in after that time will not receive any credit. So be sure to turn in whatever you have, even if it is not working fully, at that time.