

You are free to use any resources you like (e.g., papers, publicly available code, homework solutions found on the Web) in completing this homework, subject to two conditions: (1) All such resources used must be clearly and prominently acknowledged. (2) Everything you write should be original (in your own words) and you should be able to explain everything from scratch. Each question below is worth 20 points.

1. The footnote on the third page of Bloom’s seminal paper [2] includes the following comment:

If the reader wishes to analyze the comparative performance of the two methods for a machine with a multibit unit and wishes to include the effects of multiple hash codings per message, this can readily be done by following a similar method of analysis to the one used here for a single-bit unit.

Perform the suggested analysis for the following environment: Assume the hash table is stored on disk and use the standard page-based disk-access model. All items on a page are available when any one is accessed. Sequential page accesses are much faster than nonsequential ones. Assume that the available main memory is capable of holding only a small portion of the hash table at any time. Try to model the environment in a manner that best captures the parameters that are likely to determine performance. Simplify the model if you get stuck.

2. Write a program for testing the effectiveness of Bloom filters in the environment of Question 1. Perform a small experimental study that quantifies the metrics outlined in Bloom’s paper [2] as well as any others that you find interesting. Determine how well your model for Question 1 predicts the experimental results and explain any discrepancies.

In your submission, describe your experimental setup in enough detail to permit replication of your results. Submit the source code and documentation (no object code) as well. Follow standard packaging conventions. (For example, include a README file.)

3. Solve Exercises 4.1, 4.2, 4.3, and 4.22 from [1].

4. Devise a distributed join algorithm that copes gracefully with single node failures. Assume a fail-stop model, in which a node fails essentially by ceasing all operations for the duration of study (as opposed to byzantine failure modes in which it may send erroneous results). Try to minimize the amount of lost work resulting from a failure and to maximize overall performance (with and without failures). Characterize these metrics for your algorithm using the usual parameters (relation cardinalities, buffer capacities, number of processors, etc.). What is the minimum buffer space required by your method (as a function of relation cardinalities and other parameters)? Is it easy to generalize your method to cope with multiple node failures? Hint: You may wish to use the method of Example 15.18 in [3] as a starting point.

5. Write a program that takes a SQL query as input and outputs the query plan generated by PostgreSQL for that query (as a pretty-printed tree with the usual conventions). Write an analogous program for Oracle. Using appropriately populated databases, demonstrate a query and three different query plans generated by each of PostgreSQL and Oracle. Submit your packaged source code with documentation and a trace of the session demonstrating the query plans.

Submission Submit your entire homework electronically as a single gzipped tar archive. Your programs can be in any language that I can access. (Ask if in doubt.) Your text answers should be in a single PDF or plain text file. Please check that the PDF file is portable. At the very least, the `gv` program on the CSIC cluster must display it properly. Include a README file that describes the contents and their relation to the above questions. Name your file using the scheme `LastnameIJ-hw1-MNNN.tgz`, where `MNNN` is a 4-digit integer of your choice, and upload it to `ftp.cs.umd.edu` using anonymous FTP.

References

- [1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [2] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.
- [3] H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database Systems: The Complete Book*. Prentice-Hall, 2002.