Project Proposal
Asynchronous I/O and Prefetching

CS542 Database Management Systems
Instructor: Prof Elke A Rundensteiner

Yogesh Samant Vinod R Polavajram

October 9, 2003

1 Introduction

MASS, Rainbow and Vamana are projects developed at WPI. The MASS library is used by Rainbow and Vamana for storage and query execution. The MASS library has the Buffer Manager and Indexes. This project will involve changes to both. The goal of this project is to enhance the MASS Buffer Manager to enable it to do asynchronous I/O for prefetching pages from the disk storage blocks into the buffer cache. The Buffer Manager currently does all I/O synchronously. It is well-known that reading ahead using asynchronous I/O can greatly improve throughput. MASS is the right candidate for doing prefetching since query execution is generally CPU bound. Secondly it almost always performs sequential I/O.

2 Proposed Work

The primary task for this project is to add asynchronous I/O handling to the Buffer Manager. A second task would be to modify the B+ tree index so that it issues read-ahead or prefetching requests. This would require a change to the Buffer Manager interfaces that would enable it to understand and process the prefetching requests.

The motivation for doing asynchronous I/O comes from the fact that a clustered B+ tree index is used for fetching data from the disk blocks. The indexes are accessed sequentially. This means that after a request has been received for a particular page, we can subsequently pre-determine a series of pages that will be requested in sequence. This ability to predict the request pattern means that we need not read index pages sequentially in many cases. Instead, we simply make sure that the asynchronous request has completed or wait for it. We can exploit this knowledge of the sequential access method of our index and read several pages at a time instead of one page at a time. Eventually we can considerably reduce or even eliminate any waiting for synchronous I/O during query processing.

Secondly, the clustered index implies that data storage pattern on the disk is similar to the index. Consequently the Operating System would also read multiple adjacent disk blocks simultaneously. It can make the necessary optimization while reading the disk. The overall outcome will be a significant improvement in performance by minimizing the delay associated with query execution.

3 Project Architecture

The MASS library architecture will be used, since this project involves a feature addition to the MASS library itself.

A small test program will be written to exercise the new and existing features. It will test for desired functionality and expected improvement in performance.

It will also be used for functional and regression testing. Finally, if the changes work well, they will become a permanent part of the MASS library.

4 Environmental Setup

- Operating System: Linux on the CS machines at WPI.
- Language: C++
- Other: Anything else that is required to run MASS.

5 Background material

To complete this project successfully we will investigate the facilities for asynchronous I/O available in UNIX. All routines and libraries used shall conform to the POSIX standard to enable portability to any other POSIX conforming platform. Knowledge of the working of a Database Buffer Manager and indexes would also be gathered to develop a clear understanding of the tasks that need to be performed.

6 Plan of Completion

6.1 List of Tasks to be completed

1. Learn how asynchronous I/O works in Unix.
2. Obtain the MASS source code and any thing else that is needed to run it. Compile and run it on any of the CS Linux machines at WPI.
3. Write a small program that exercises the Buffer Manager interfaces – Create, Release, Find, Destroy, so that we understand the existing functionality.
4. Create a simple database structure for testing prefetching.
5. Add asynchronous I/O to the Buffer Cache. This would require changes to the four interfaces listed above and make them prefetch-capable.
6. Test it using our simple database structure.

6.2 Additional Tasks that may be completed

1. Modify the B+ Tree index so that it issues read-ahead requests using the prefetch-capable interfaces of the Buffer Manager.

More tasks may be added if required. Tasks 1-6 should be completed within 4 weeks, beginning today. If tasks 1-6 are completed in a timely fashion and no additional tasks are added later, task 7 will be taken up. It should be completed within the next 2 weeks. The last 1 week will be reserved for functional and regression testing.

The distribution of work between the two of us has not been decided.

7 Deliverables

The final deliverable for the project will be the MASS source code with the additional capability of accepting prefetch requests and performing asynchronous I/O. In addition we would also deliver any test program that would be used to test the functionality. Finally, if everything goes well, we would also aim to provide documentation for the enhancements made to the MASS library.