Virtual Memory Alternatives for Transaction Buffer Management in a Single-Level Store

by Dylan McNamee, Vivek Narasayya,
Ashutosh Tiwary, and Henry Levy
University of Washington

Jeffrey Chase and Yong Gao
Duke University

Motivation

Until recently, database systems primarily ran on dedicated server machines. Database systems are now often run outside of the dedicated server environment: on client machines, personal computers on user desktops, or multi-use enterprise servers. The key difference between these environments is that dedicated server machines run the database system in isolation, while in the other environments, the database system shares the machine with other applications (e.g. document processors, email programs, web browsers, and perhaps other database systems). In these environments, the database system needs to cooperate with the operating system to effectively share the physical memory of the machine with the other applications on the machine. We are particularly interested in supporting applications that perform transactions in a single-level storage environment (e.g., clients of current object-oriented databases).

In current commercial operating systems, virtual memory and database buffer managers are oblivious to each other. Thus, when virtual memory paging occurs, a database buffer manager can incur double paging as it attempts to access its buffer memory.

Previous operating systems research has addressed this problem by providing database memory management functionality within the operating system (e.g., TABS, BubbaOS, Camelot, Cricket, RPVM). This approach solves the problem by integrating database buffer management with virtual memory. This solution allows databases to achieve excellent performance while allowing memory to be effectively utilized by all of the applications on the machine. Taking advantage of these results has not been feasible for commercial database designers, however, because these solutions require features of research operating systems that would require significant redesign of commercial operating systems to implement. Further, this platform mismatch has made it difficult to project the performance results of these systems to commercial operating systems.

The Work in Progress

The work in progress we will describe defines a space of trade-offs between database performance in a multiprogrammed environment and the required degree of operating system support. We argue that the current alternatives represent the two possible extremes: poor performance with no additional support required, and excellent performance with extensive operating system support required.

Second, we are investigating two new points in the space of trade-offs that lay between these extremes. Both of these techniques improve performance over current practice by utilizing existing operating system facilities to integrate database buffer management with operating system virtual memory management. The first approach manages transaction logging on the client, the second approach performs shadow-paging on the server.

Third, we describe an implementation of the new approaches in a commercial operating system, Digital Unix, running on AlphaStation 250 workstations. We also present two more systems, built on the same platform, that are representative of the two existing alternatives. The implementation of the four systems on an identical platform enables, for the first time, a fair evaluation of the full spectrum of degrees of virtual memory support for database buffer management.

We conclude that current practice of transaction buffer management performs poorly in the increasingly common memory-competitive environments. We argue that this poor performance is the result of fundamental structuring choices. We will lend evidence to this argument by demonstrating significant performance improvements in a restructured system that uses existing operating systems facilities.