On using Reliable Network RAM for Database Systems

Athanasios E. Papathanasiou and Evangelos P. Markatos *

Institute of Computer Science (ICS)
Foundation for Research & Technology – Hellas (FORTH),
P.O.Box 1385 Heraklion, Crete, GR-711-10 GREECE
{papathan, markatos}@ics.forth.gr
http://www.ics.forth.gr/proj/avg/paging.html

The performance of traditional database systems has been limited by the latency of magnetic disks, where they store their data. Magnetic disks have an access time, which is orders of magnitude higher than that of main memory storage. Thus, they impose a significant overhead in transaction processing systems. The need for better response times and greater transaction throughput makes it necessary to remove the access of magnetic disks from the critical path of the transaction processing.

To ensure recovery in case of failures, most database systems use a (write ahead) log file. The log file must reside in stable storage (usually magnetic disks). Accesses to the log file are usually in the critical path of the transaction processing and usually need synchronous input/output (Figure 1).

In this project the main memory (DRAM) of a network of workstations is used as a form of reliable memory. By storing sensitive data in the main memory of more than one workstations (connected to different power supplies) we are able to recover them in case one workstation fails. Our goal is to improve the transaction throughput and the response times of transaction processing.

In our first design (Figure 2) we use network DRAM to create a layer of reliable (recoverable) memory, which completely eliminates the need of a log file. The whole database is mapped to the main memory of remote nodes. When a transaction begins, the application notifies the transactional library of the portion of the database that is going to be updated. The transactional library creates an undo log (a copy of the original data) in both the local and the remote memory. Then, the application executes the transaction. After the completion of the udate operation, the modified portion of the database is copied to the remote node, and the transaction commits. Since data, modified by commited transactions, exist in the main memories of two nodes, they may be recovered in case of a single workstation failure. The undo logs are kept in the remote memory for as long as the transaction executes. When the transaction commits the space of the undo logs is reclaimed.

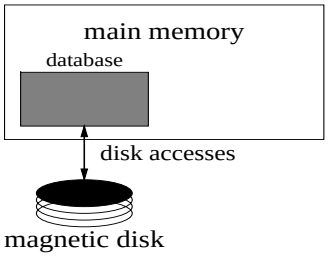

Figure 1: **Traditional DB Systems.**

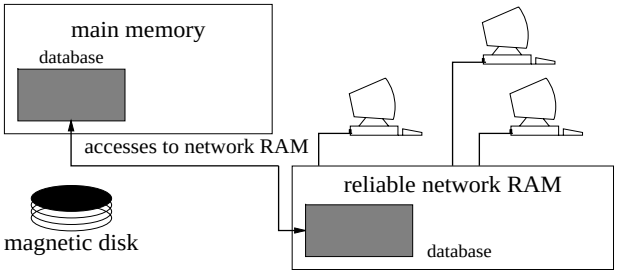

Figure 2: **Our approach.**

In our second approach, we use a traditional (redo) log file. Transactions commmit based on the Write Ahead Logging Protocol. However, in our implementation the (redo) log is kept in reliable network DRAM, instead of magnetic disks. Thus, synchronous accesses to the (redo) log do not invoke synchronous disk accesses, but synchronous network DRAM accesses, which are generally faster. When several transactions have commited, the changes propagate from the redo log to the on-disk copy of the database asynchronously.

We have started implementing our transactional library. Our experimental platform consists of two PCs running WindowsNT, connected with two Dolphin PCI-SCI (Scalable Coherent Interface) network cards.

*The authors are also with the University of Crete