SHORE: Combining the Best Features of OODBMS and File Systems*

The SHORE Team
Computer Sciences Department,
University of Wisconsin, Madison
shore@cs.wisc.edu

1 Introduction

Shore is a persistent object system under development at the University of Wisconsin that represents a merger of key features of object-oriented database (OODB) and file system technologies: Concurrency control and recovery and bulk operations on clustered data as in traditional databases; objects of widely varying sizes with individual identity, globally unique identifiers, and a rich object-oriented type system as in OODB systems; and an open architecture, a uniform, universal hierarchical name space, and controlled sharing of disks with reliably enforced ownership of data and space accounting. A more complete overview of the goals and structure of Share were presented at this conference last year [Carey *et al*, SIGMOD 94].

2 The Architecture of Shore

Shore consists of a *server process* running on each host and a *client library* linked into each application program. The client library maintains an LRU cache, fetching objects on demand and swizzling references. The core of the server consists of a *Storage Manager*, which controls local disks (if any) and acts as a page cache, fetching pages from peer servers on behalf of local application processes. It implements transactions, concurrency control and recovery with an integrated locking and cache consistency protocol [Carey, Franklin, Zaharioudakis, SIGMOD 94].

Additional features are provided by *Value-Added Servers* (VASs). The server architecture is layered so that special-purpose server code can be linked with the Storage Manager, gaining direct access to its data structures and threads package, subject only to the visibility controls dictated by the C++ language. The "Shore VAS" supports a Unix-like namespace and protection model. It communicates with client processes through a special-purpose protocol, using shared memory if possible to speed up bulk transfer of data, and using Unix address-space separation to protect the integrity of key data structures from buggy or malicious clients. Another VAS, layered on top of the Shore VAS, responds to Sun Network File System (NFS) requests. With this facility, users can "mount" portions of a Shore database into a standard Unix directory hierarchy and access certain Shore objects as if they were ordinary Unix directories and files, allowing the vast body of application software written for a Unix environment to be used unchanged on top of Shore. The VAS architecture also allows applications to bypass layers of software where appropriate, as illustrated by the Paradise GIS system (described elsewhere in this proceedings), which has a VAS that despenses with the Unix namespace and type system provided by the Shore VAS, and uses physical rather than logical OIDs.

3 The Shore Data Model

The Storage Manager supports *objects*, variable-sized untyped arrays of bytes allocated out of clusters called *stores*, and various kinds of index structures. The Shore VAS associates with each object a set of *system properties* such as ownership and protections, and a pointer to a "type object." It also distiguishes certain kinds of objects, including *directories*, *symbolic links*, and *pools* (which correspond to stores). The Shore VAS does not distinguish between other objects (they are like Unix "plain files"), but it does allow each object to designate a portion of its state as the "Unix text" field, which is visible through the NFS interface. The directory hierarchy is fully compatible with Unix semantics. In particular, it ensures that all allocated space can be associated with pathnames in the directory name space.

Type objects are described in the *Shore Data Language* (SDL), a dialect of ODMG's ODL [Cattell, ODMG-93.1, Morgan Kaufman]. SDL specifications are compiled into type objects stored in the database, from which language-specific bindings are generated. A C++ binding includes class definitions and support routines that allow objects that are instances of SDL types to be manipulated in memory as instances of corresponding C++ classes. Other lanaguage bindings are planned. There are also facilities for browsing objects by fetching and interpreting the corresponding type objects.

4 The Demonstration

We first demonstrate Unix compatibility by building other parts of the demonstration with unmodified utilities (such as make, cc, ld, etc.) and all source and object "files" stored as Shore objects. We also demonstrate a facility to wrap Unix commands in serializable transactions. We then run parts of the oo7 benchmark [Carey *et al*, SIGMOD 93] and illustrate browsing of oo7 structures through a Mosaic interface. Finally, we demonstrate an information-retrieval application implemented on Shore, which maintains an inverted index on the Shore documentation. Shore and Paradise demonstrations will alternate.

*This research is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518.