Architecture of the Encina Distributed Transaction Processing Family

Mark Sherman
Transarc Corporation
707 Grant Street
Pittsburgh, PA 15219
mss@transarc.com

Abstract

This paper discusses how the Encina® family of distributed transaction processing software can be used to build reliable, distributed applications. We start with the toolkit components of Encina and how they are used for implementing ACID properties. We then consider how the toolkit can be applied in building higher level components in a DCE environment. We conclude with a discussion of the Encina Monitor, which provides a framework for organizing a collection of machines and servers.

1: Introduction

The Encina family of transaction processing software provides a commercial implementation of advanced transaction processing research. Many of the ideas embodied in Encina are an outgrowth of the TABS and Camelot research projects [1], which in turn were adjuncts to the Mach project, all at Carnegie Mellon University. Key features of all of these projects include: they were designed from the outset to be used in a distributed environment; they were built to be extensible; they were built to have replaceable components. Encina embraces the same goals: distribution, extensibility and module replaceability. Hence, we first designed and implemented a collection of software components that provide tools for building distributed transactional systems. We used those components to build prerequisite services that enable the construction and administration of large-scale OLTP systems. This paper first discusses the lower level components and their capabilities that provide the core of Encina. The discussion continues with some of the higher-level services built on those components. This discussion assumes a basic knowledge of transaction processing systems as in [2].

2: Structure of Encina

Encina is a layered, modular collection of software components, shown in Figure 1.

SIGMOD /5/93/Washington, DC,USA
© 1993 ACM 0-89791-592-5/93/0005/0460...$1.50

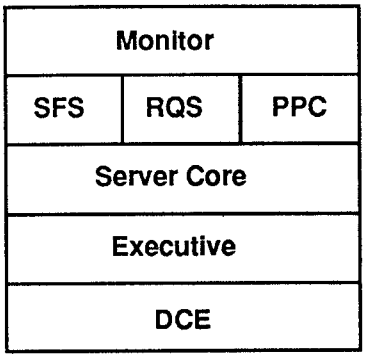

Figure 1: Encina Architecture

We start with a collection of basic distributed services called the DCE, which is the Open Software Foundation's Distributed Computing Environment. We extend the DCE with a collection of basic transaction services called the Executive. The Server Core provides additional capabilities for defining and managing recoverable storage. Together, the Executive and the Server Core compromise our Toolkit for building transactional components. Encina also includes two resource managers built on top of the Toolkit: the Structured File Server (SFS) and the Recoverable Queueing Server (RQS). The Peer-to-Peer Communications (PPC) manager is also built on top of the Toolkit. In its current release, Encina also has a monitor provided, which simplifies the creation and administration of a large-scale distributed system. In the rest of this paper, we discuss each component.

3: Extending DCE Basic Services

Any distributed system requires basic services, such as participant authentication, secure communications, a way to locate objects (e.g., servers, machines) in an environment and concurrency support. We chose to use the OSF's DCE as a basis since we believed that the DCE will be ubiquitous.

The DCE provides some key services for Encina. First, Encina uses DCE's threads, which provide the ability to perform efficiently multiple tasks simultaneously. Encina extends the thread system of the DCE in two ways. The first addition is structured use of threads. DCE threads use a coroutine-based model, that is, a thread is a coroutine started on a new stack. While this permits a large degree

of language independence, the approach also offers no linguistic support for writing concurrent programs. The Transactional-C facility of Encina extends the C language with two control features on top of DCE threads: a concurrent **for** statement and a multiway **concurrent** (fork-and-join) statement. The second extension is the ThreadTid facility. This module is used to coordinate transaction state with individual DCE threads, allowing a single Unix process to work on more than one transaction at a time.

The second DCE feature used by Encina is the directory service. The directory service in the DCE uses the Unix file name-space as a paradigm for organizing information about a collection of machines. All resources are described by a large tree structure which looks like a Unix file tree. Some parts of the tree do represent Posix files — specifically the part used by the DFS (distributed file system). The DCE defines other parts of tree to describe other information, such as principals (e.g., people and programs). Encina extends the tree structure both to locate servers for a client and as a framework for organizing files in the SFS, RQS and Log.

Third, the DCE security services provide Encina with the ability to authenticate a participant and define a participant's authorization to a server. Encina extends the DCE services by allowing administrators to set authorizations on servers just like one sets access protections on files. This simplifies the ability to specify who is allowed to access a service.

Last, but not least, Encina uses DCE's RPC mechanism as the basic communication paradigm between clients and servers. Encina extends the RCP mechanism to make it transactional. The extended mechanism, called TRPC for Transactional RPC, allows one to make RPC calls without explicit in-line checking for failures. The very nature of transactions provide an all-or-nothing semantics for executing a collection of operations. In a TRPC environment, this means that all RPCs to servers either successfully completed or that all participants in the transaction will be automatically notified of an error anywhere in the transaction and rolled back.

4: Executive

The Executive part of the Toolkit provides the basic transaction description and communication facilities. The most fundamental facility is the distributed transaction service (TRAN). Like all transaction managers, TRAN provides transaction demarcation and two-phase commit coordination. TRAN is unique in that it provides a nested transaction model and separate interfaces for communications systems, logging systems and recovery systems. Another facility is a language extension to C called Transactional-C. As mentioned earlier, Transactional-C provides structured threads to DCE. But it goes farther by providing integrated support for nested transactions and transactional memory management in a multi-threaded environment. We implemented Transactional-C as macros to ANSI-C to permit use with standard compilers.

5: Server Core

The Encina Sever Core provides facilities for managing recoverable storage. There are two collections of such facilities in the server core. The first collection can be used for building new kinds of recoverable servers, such as databases. The first collection contains four parts. The first part is the volume service, which provides an abstraction of a highly reliable, large disk. Among the features provided by the volume service are automatic mirroring of disks and support for files that span multiple disks.

The second part is a log facility, which provides a shared logging server. The log can be viewed as another DCE file system. To the administrator, another subtree of the DCE name space contains the names of log servers and their associated log files. Programmers open a log file by passing an appropriate path name, much like Posix files. Of course, the semantics and operations of a log file are different than for a Posix file. For example, one may only write at the end of the file, and log files typically grow without bound. File archiving is provided by backing up the head of the file separately from the rest.

The third part of the server core used for building recoverable storage is a recovery manager. The recovery manager provides the abstraction of a recoverable page of memory. Programs bring in a buffer, perform updates and then release the page. Committed changes are permanent — they survive system crashes. Aborted changes are automatically reversed. This is accomplished by logging changes, and performing the necessary redo and undo operations. Normally, changes to the buffer are automatically logged and consist of either physical changes to the page (before and after images of the page contents) or logical changes to the page (operations that can be replayed to either recreate the changes or undo the changes). The recovery service handles all physical memory management including cache management between the disk and in memory versions of the pages. Backup of the pages is supported by a fuzzy-dump system that allows on-line backups to be performed on running systems.

The fourth part is a lock manager that supports a wide range of locks for serialization. These logical locks come with a variety of conflict semantics that support both

conventional read, write and upgrade locks, as well as hierarchical locking and range locking. Like TRAN, the lock system supports nested transactions.

The second collection of facilities for managing recoverable storage implements the X/Open XA interface. The XA interface permits non-Encina resource managers to be integrated into an Encina-based distributed transaction system.

6: Resource managers

We have built two resource managers on top of the Toolkit, i.e., Server Core and Executive. Because these resource managers inherit the capabilities of the Toolkit, they support distributed updates, nested transactions, multithreaded access, on-line fuzzy dump backups and DCE security.

The first resource manager, the SFS, provides a record-oriented, transactional file system. The SFS can be used where the Unix file system semantics are insufficient for failure recovery or concurrency control, or when a record structure is desired instead of a stream of bytes. Following the DCE model, SFS servers and SFS files appear to be subtrees in the DCE name space. Programmers access these files by opening a path name, making record operations and closing the file. Similarly, permissions on the files are set using ordinary file access control lists, making the system very natural to administer.

The second resource manager, the RQS, provides a stable intermediate data structure for serial transactions. A transaction may use queues to time-shift part of the transaction, sort collections of transactions, break long running transactions into smaller chunks or process certain kinds of failures. Like servers and files in the SFS, RQS servers and queues look like another subtree in the DCE name space, making them easy to manipulate and administer.

7: Communication managers

Encina provides two kinds of communication facilities for distributing transactions among participants: DCE RPC and PPC. DCE RPC, along with its transactional extensions in TRPC, has been previously discussed.

PPC (peer-to-peer communications) provides a connection-oriented, stateful communications paradigm that is commonly used on existing mainframe systems. Encina's PPC provides this facility between two machines operating in a DCE environment, and between a DCE-based machine and a mainframe. In the later case, the communication is performed over a LU6.2/SNA connection. The result is that the Encina system looks just like another LU6.2 mainframe host, including two-phase commit processing (called "syncpoint" in mainframe terms).

8: Monitors

Historically, a transaction "monitor" encompassed everything that has been discussed so far. In the Encina architecture, a monitor provides additional execution and administrative functions for a collection of clients, servers and resource managers. Encina has been designed to allow for a number of monitors to be used and interoperate.

The Encina Monitor within the Encina family provides a "3 box" model of system organization. The first box represents clients of the system. Typically, clients gather information from a user, process gestures, perform some local validation of data and run graphical interfaces. Once a request is specified in a client, the second box, an application server, comes into play. A client calls the application server to process the request. The application server embodies the processing required for a business function. For example, an application server may provide the "reserve a hotel room" service. The application server would check to make sure that the requester is permitted to make a reservation, check for empty rooms and ensure that the credit card for the reservation had available credit. To carry out its processing, the application server uses the third type of box, resource managers. Resource managers are typically databases and hold information. For example, the application server might check one database for room availability and another for credit authorization. Of course, resource managers might not be data, but could be communication resources. The application server might use some kind of communication to a credit card clearing house to get the necessary credit validation. The isolation of the business function into the application server permits implementation to use (and to change) the resources it needs to implement a desired service.

In addition to supporting the 3 box model of system organization, the Encina Monitor, which distributes clients among equivalent servers for load balancing, automatically applies DCE authorization mechanisms to application servers, provides a logical operator's console across a collection of machines, configures a collection of machines and servers, as well as a host of other services. The Encina Monitor uses a DCE-like style of administering a collection of machines and servers.

9: Summary

This paper has briefly described the pieces of Encina. Together, the components provide a complete and modern distributed transaction processing system. However, the

modular decomposition of the software allows pieces to be used and extended in a number of ways. Thus, Encina is not only an OLTP system — it provides the necessary supplements to the DCE to make distributed application creation easier and more reliable.

References

[1] *Camelot and Avalon: A Distributed Transaction Facility*, Jeffrey L. Eppinger, Lily B. Mummert and Alfred Z. Spector editors, Morgan Kaufmann Publishers, 1991.

[2] Jim Gray and Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann Publishers, 1992.