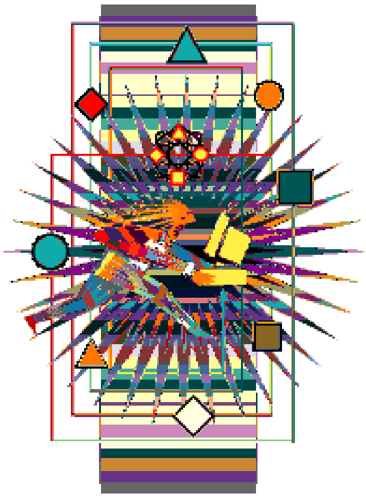
Articles & White Papers

ON MIDDLEWARE

By Max Dolgicer



Only the TP monitor manages integrity across database mix

As corporations have started to tackle more serious client/server applications, the notion of middleware has been cemented in the minds of corporate developers. Everyone from enterprise architects to business application developers agrees that middleware is a "must have component" for any serious client/server application. However, not too many agree on which kind of middleware should be used, and when.

One of the most debated issues is if and when one of the more mature categories of middleware such as portable, distributed transaction processing (TP) monitors should be used. TP monitors are complex subsystems that provide a variety of middleware services, including communication, administration facilities, multithreading and load balancing, among others. Portable TP monitors include products such as Tuxedo from Novell Inc., Provo, Utah, which was recently turned over to BEA Systems Inc., Sunnyvale, Calif.; Encina from Transarc Corp., Pittsburgh; Customer Information Control System (CICS) from IBM; Top End from NCR Corp., Dayton, Ohio, which was formerly AT&T's Global Information Solutions; and UniKix from UniKix Technologies, Billerica, Mass.

Originally, this debate was known as TP-Lite versus TP-Heavy, where the former was almost always represented by two-tier applications, where there were no application servers, and where the graphical user interface (GUI) front end communicated directly to the database engine, which functioned as a transaction manager.

Today the debate is much more complicated. Three-tier TP-Lite has emerged, where an application is partitioned into a GUI front end, an application server and a database server.

From a transaction management point of view, the three-tier TP-Lite model still implies that the database alone manages the transactional context and is the transaction manager that your application relies on exclusively for data integrity and recovery. But, three-tier TP-Lite also implies that different categories of middleware, such as object request brokers (ORBs) based on the Common Object Request Broker Architecture (Corba) of the Object Management Group (OMG), Framingham, Mass.; messaging; Distributed Computing Environment (DCE) from Open Software Foundation, Cambridge, Mass.; and the forthcoming Network-OLE (N-OLE), which was recently renamed Distributed COM (DCOM), from Microsoft Corp., could be used for these applications.

So what is three-tier TP-Heavy? At least to date, three-tier TP-Heavy refers to TP monitors used as the middleware application. Transaction monitors are not just a middleware glue that holds different application tiers together (that is, allows clients to communicate with the application servers); they also partake in transaction management, where they take direct responsibility from a database engine to coordinate and manage the transactions.

In the future, other types of middleware could acquire transaction management capabilities. For example, some Corba products are expected to implement what is called Object Transaction Service (OTS), which has been defined by the OMG as a part of the Corba 2 Common Object System Services (COSS) specifications. By doing so, Corba ORBs will be able to provide transaction management capabilities. However, as of this writing, none of the Corba products being shipped today have these capabilities.

The Basics

First, in a distributed environment, updates to multiple resource managers (that is, databases) have to be coordinated across all platforms. One of the ways to achieve such coordination is through a two-phase commit protocol (2PC). All major relational database management system (RDBMS) vendors have implemented 2PC. Furthermore, transactional semantics (begin transaction, commit transaction, rollback transaction), including 2PC, can be performed by an RDBMS as long as you are running the same RDBMS at all sites.

But what happens if your application attempts to update heterogeneous RDBMSs in one transaction? Consider this scenario. Your transaction updates database tables from Informix Software Inc., Menlo Park, Calif., and Sybase Inc., Emeryville, Calif., with an absence of an intermediary capable of performing transaction management. Who should be responsible for coordinating these updates and guaranteeing data integrity across multiple databases? Informix? Sybase? I don’t think so.

Such a feature is unique to TP monitors. In other words, there is no other middleware technology capable of coordinating and managing transactions across heterogeneous databases. Furthermore, such a coordination is in line with the Distributed Transaction Processing (DTP) standard published by X/Open Company Ltd., headquartered in the U.K., but with U.S. offices in Menlo Park, Calif. The XA protocol specifications published by X/Open have defined a standard interface for communication between a transaction manager and a resource manager. Today all portable TP monitors and all major database vendors are XA-compliant. Adherence to this common protocol allows TP monitors to update heterogeneous databases and at the same time preserve their integrity within the transaction.

TP monitors also preserve a transactional context across multiple processes that can be executed and update multiple databases under the same transactional umbrella. Let's assume that one process updates a database and transfers control to another process, which in turn updates another database. To assure that all updates will either commit or rollback, software capable of retaining transactional context across separate processes is required. To date, and to the best of our knowledge, only TP monitors are capable of doing so.

ACCESS TO NON-RDBMS DATA SOURCES -- Today all client/server applications are using RDBMSs. Sometimes RDBMSs are used as a storage medium, due to the inability of development tools to access anything but RDBMSs. TP monitors are capable of accessing not just RDBMSs, but also record-oriented files. This allows corporate developers to use RDBMSs and pay license fees to respective vendors only when advantages provided by a relational model are really utilized.

SECURITY -- Generally, TP monitor products support different levels of authentication. Some products also support data encryption and authorization, which allow the securing of specific business functions by utilizing access control lists (ACLs). For example, Encina inherits and enhances DCE's Kerberos mechanism (this also applies to Top End, which has integrated Kerberos in the product), while Tuxedo has its own security mechanism and is planning to support the Kerberos mechanism in the future.

Some middleware products provide their own level of security in addition to the security provided by the operating system that they are running on. However, the majority of other middleware products do not deal with security issues yet, which basically leaves you with three choices. You can either implement security from within your application, rely on the operating system's security, or integrate the middleware product of your choice with an external security product. To rely just on the operating system's security, especially when the operating system is, for example, Unix, might not be such a great idea.

SCALABILITY -- There is no question that a three-tier application model will scale better than a two-tier application model. One of the reasons is the fact that with three-tier, an application can be partitioned into user interface, application server and database server. In order to accommodate a larger number of users without compromising performance, additional application servers can be added as necessary. It should be noted that application partitioning can also be supported by many middleware product categories, and not just by TP monitors. ORBs, messaging and DCE are among the middleware products that support three-tier application architectures.

Scalability is a function of many variables (efficient and flexible communication between different application parts, load balancing, instance management and so on, are just a few examples of these variables), and to say that TP monitors always provide better scalability than other middleware categories is not correct. A lot is dependent on the specific middleware product itself and the business application. However, all TP monitors -- without any exceptions -- have been specifically designed from the onset to accommodate many concurrent users and to handle many database connections. These TP monitors can all handle high transaction volume without compromising performance, as well as accommodate additional users without major reconfiguration.

ASYNCHRONOUS PROCESSING -- Some TP monitors support an asynchronous, nonblocking communication. In other words, a client is not blocked while the server is formulating the answer. In addition, all major TP monitors support asynchronous, queue-based operations, where a request can be queued-up or deferred for subsequent processing. This provides time independence, that is, different application components, or even independent applications, can run independently of each other. Again, almost all major message-oriented middleware (MOM) products support this feature. Support for this feature by additional middleware categories (for example, database access middleware) has already been announced.

EFFICIENCY -- TP monitors reduce CPU overhead, response times and CPU cost for large applications by having the ability to dispatch, schedule and prioritize multiple application requests concurrently. Internally at least, some TP monitors are utilizing very efficient operating system primitives such as Unix queues or shared memory. This feature can also be found in some other types of middleware.

DYNAMIC RECONFIGURATION -- TP monitors permit the dynamic reconfiguration of the environment, including transactions and database table names, without forcing the need to bring the system down. Some TP monitors reduce administrative complexity by providing their own administration facilities and runtime statistics. This feature alone allows the ability to keep track of resource utilization. However, some other middleware products, for example the Pipes Platform from PeerLogic Inc., San Francisco, provide support for such features as well.

LOCATION TRANSPARENCY -- TP monitors provide different levels of naming and directory services. Some of them offer the concept of cells or domains. (Cells or domains are simply a set of machines and users under common security and administrative authority.) Each of the service providers in a cell or domain, including business applications, must register their current physical location and other information in a directory service using a logical name. Such a directory service can be a standard one provided by DCE, as in the case of Encina, or a proprietary one, such as the one provided by Tuxedo or Top End. Support for naming services among other middleware products ranges from none (that is, no naming service) to a very sophisticated one (for example, SuiteDome from SuiteSoftware, Anaheim, Calif. and PeerLogic's Pipes Platform).

LOAD BALANCING -- When it comes to load balancing, TP monitors support a number of sophisticated features. For example, in order to deliver consistent response times, they are capable of starting additional instances of end-user processes. Generally, load balancing criteria can be configurable and can range from a simple round-robin algorithm to sophisticated configurable thresholds where numbers of outstanding requests would start another instance of the process. At least for now, this feature (as well as the next one) is quite unique to TP monitors.

RELIABILITY -- Most TP monitors support automatic fail-over and certain ones posses restart capabilities, thus increasing application up-time.

PERFORMANCE -- TP monitors are tuned to accommodate a large number of users concurrently and use computer resources more efficiently. Most of them have an embedded logic to share database connections among many transactions. With TP monitors, the number of operating system processes doesn't go up exponentially with the number of users, thus avoiding the need for the operating system to deal with as many processes. This feature can boost the number of concurrent users substantially without compromising on the response time.

Some multithreaded databases such as Oracle 7.x and Informix 5.x do not require as many processes as they used to. However, in the three-tier environment, a communication overhead between the client nodes and the database servers can be reduced by the presence of a middle tier (that is, an application server) capable of multiplexing network connections, as well as reusing them. This can be done by a number of other types of middleware as well.

The Bottom Line

TP monitors definitely should be used only when your application or enterprise architecture requires features that are capable of being fulfilled by TP monitors only. If this is not the case, TP monitors will continue to compete with other types of middleware. This is true because the complexity of the technology, combined with the cost and overhead, would be overkill for most situations, thereby forcing the decision makers to ask themselves whether TP monitors are warranted and justified.1