5 Distributed computing model

The two major distributed computing models available today are CORBA, developed by a non-profit consortium of vendors and users, and Microsoft's Distributed Component Object Model (DCOM).

Currently DCOM can only be deployed on PC platforms, while CORBA can be used on either PC or Unix platforms. Typically, the CORBA model is more open because of platforms and languages supported, but ORB vendors are beginning to provide CORBA to DCOM bridges.

5.1 Development language

The development language is important on two accounts. First, it may determines the application server platform. Second, it affects your choice for the appropriate distributed computing model.

For example, if the development language is Java and the platform is from Sun, the user has a choice of NetDynamics, Netscape Application Server, GemStone, and others. If the language is Visual Basic, the platform choice is a PC and Microsoft Transaction Service (MTS) naturally has an advantage. Generally, using Java as the development language offers the highest level of platform independence.

5.1.1 Platform

If you start the decision-making process at the platform level, you'll find that Unix platforms tend to scale better than the PC. Microsoft is working hard to change this over time, with coming offerings such as Windows 2000.

5.1.2 Achieving scalability across one or more systems:

The scalability issue applies within a single process, across multiple processes or across multiple systems. For an application to scale within a process, the run time has to be multithreaded or at least thread safe. The infrastructure to support multithreaded applications is normally part of the application server. Once an application is launched,

the process environment—multithreaded or single-threaded—is determined from a configuration setup.

When scaling across multiple processes, the application server should ensure that applications can be assigned to a free process. If no free process is available, the application server should queue the request or initiate other processes.

The more difficult issue is whether or not an application can scale to another process in another system. The Netscape Application Server, for example, views processes running on multiple systems as a single entity. Through configuration data, an application can run on any process on any system within the configuration. The application server lets you add systems to the environment, rather than continuously upgrading a single system. Ultimately, there is a limit to the size a single system can be upgraded, and true scalability can only come from adding systems.

5.1.3 Services needed for application server environment

Application servers rely on a set of interfaces to invoke services such as database access, security, transaction integrity, and so forth. Look carefully at services before deploying an application server. For example, most offer basic services but may not provide transactional services. Some servers allow you to write your own services, using adapters or extensions.