Three-tier client server architecture: achieving scalability, performance, and efficiency in client server applications
Wayne W. Eckerson
***Open Information Systems.*** 10.1 (Jan. 1995): p3. From *General OneFile*.

Full Text:

Achieving Scalability, Performance, and Efficiency in Client/Server Applications

Introduction: Setting the Stage

Architecture Is Critical

The most hardened mainframe bigot has come to realize that client/server computing is not just another passing fad in the computer industry. Client/server computing is here to stay. The challenge now facing users is not whether to adopt client/server computing but how to make it work effectively to solve business problems on an enterprise scale.

Most client/server applications today are departmental in scope. They are built on a two-tier architecture that is designed to support 15 to 20 users and run non-mission-critical functions. Most of these applications have been built with popular graphical user interface (GUI) development tools that pack all the code for the user interface, application logic, and services in a Windows PC. The client issues SQL calls across a local area network to a relational database to retrieve data for processing.

Encouraged by the successes of these initial client/server applications, companies have been quick to expand the scope and complexity of the applications to address pressing business opportunities or requirements. Managers have asked developers to increase the number of users, functions, and data sources supported by these applications. This inexorable "scope creep" has been the downfall of many early client/server applications.

Developers are now recognizing that a two-tier architecture is not sufficient to support enterprise client/server applications that provide high performance, scalability, availability, and reliability.

Middle Tier of

Application Servers

THREE-TIER MODEL. Enter the three-tier client/server architecture. The three-tier architecture extends the previous model by adding a middle tier of intermediate servers to support application logic and distributed computing services. The middle tier is critical for providing location and migration transparency of distributed resources as well as a variety of services required to provide reliable, secure, and efficient distributed computing.

Companies planning to build more scalable client/server applications should consider deploying a three-tier architecture. A three-tier architecture will allow introduction: Setting the Stage companies to leverage their investments in client/server to create applications that provide real bottom-line payback.

However, building three-tier applications isn't easy, and there's little real-world experience to shed

design and deployment of three-tier applications, but these tools are still immature and don't provide all the services needed to support a distributed computing environment.

This article analyzes the makeup of a three-tier client/server architecture and examine the benefits and challenges it offers. It also examines the tools available today for building three-tier client/server applications and the key features and services they should support. Finally, the report examines how three-tier client/server architecture lays the foundation for distributed object computing.

Definition of Terms

Application Components

vs. Physical Platforms

As with any new buzzword in the computer industry, three-tier architecture means slightly different things to different people. The precise definition of tiers and what they represent in user computing environments are fuzzy.

A three-tier architecture has two aspects. One is the application architecture, and the other is the deployment architecture. Application architecture refers to the constituent parts that make up an application. These are presentation, functional logic and data. Together, these components comprise the logical representation of an application.

A deployment architecture, on the other hand, specifies the physical configuration of computers on which an application runs. In a two-tier deployment architecture, the client portion of the application runs on a desktop PC or workstation, and the server portion runs on a server machine across a network. The workstations and server machines are considered the tiers in a deployment architecture.

A three-tier deployment architecture inserts an intermediate server between the desktop PC and the central server. In a three-tier architecture, the desktop PC handles the application's presentation component, the intermediate server supports functional logic and services, and the back-end server handles data processing. Here the physical deployment environment mirrors the logical application architecture. This creates an inherent synergy that provides many benefits. (See Illustration 1.)

Application Architecture

Application Components

As was mentioned above, the three basic elements of an application are presentation, functional logic, and data. Presentation refers to the user interface, or the part of an application that controls what is displayed on a user's workstation screen. Functional logic refers to the tasks and rules that govern the way a business operates and the services required to implement them. Business rules are embedded in programming code. Data refers to the information the business accumulates about its customers and operations as well as the services required to access and manage those data.

DECOMPOSABILITY. The key to three-tier application architecture is to ensure that each of these

say this is that three-tier applications can be decomposed into presentation, functional logic, and data. This means that each component can be modified or replaced without any of the other components being rewritten or changed.

Many Applications Aren't

Decomposable

Most older mainframe applications and some early client/server applications are not decomposable. Instead of being logically separated, these components are inextricably intertwined in the fabric of the application. Consequently, application components can't be easily extracted and ported to another platform or changed without affecting other parts of the application. It is difficult, for example, to migrate these types of mainframe applications to client/server platforms. Either the code is ported in entirety to Unix, or the applications have to be rewritten from scratch.

GUI Builders. Client/server applications developed with popular GUI builders, such as Powersoft's Powerbuilder, link presentation and logic and rely on a remote server to store and manage data. These tools let developers paint application screens with GUI objects, such as radio buttons and dialog boxes, and then attach scripts to each object. Clicking on the object activates the script, creating an event-driven program. The logic in these applications is tied to the GUI. Changing the logic requires rewriting a script and relinking the script to the appropriate GUI object.

In a decomposable application, presentation and function are separate, independent entities. This allows developers to run presentation on a desktop PC and functional logic on one or more computers on the network, which makes it easier to update applications since the logic is located in one place instead of on hundreds of remote computers.

Application Interfaces

There are three ways to create decomposable applications. One is to use a traditional programming language to segment monolithic applications into callable procedures. Another is to link programs running presentation, logic, and data components via common interfaces, such as remote procedure calls (RPCs) or application programming interfaces (APIS). The third way is to build presentation, function, and data components from a series of autonomous objects. (See Illustration 2). Application interfaces can range from SQL APIs, such as Microsoft's Open Database Connectivity (ODBC) and Sybase's DB-Lib, to RPCs to vendor-proprietary APIs.

Building interfaces

between All Application

Components May Be

Overkill

For example, middleware vendor Open Environment has helped evangelize the notion of three-tier architecture. Its tools enable developers to build interfaces between presentation, function, and data service components of an application using a proprietary RPC or the Open Software Foundation's (OSF's) Distributed Computing Environment (DCE) RPC. However, building RPC interfaces between all application components may be overkill for many client/server applications.

numbers of clients need access to lots of distributed services in order to execute applications.

Hybrid interfaces. A more common approach is to leverage vendor interfaces to link up components that may be distributed across platforms. For example, 4GL vendor Magna provides a transactional API on Windows PCs that enables developers to integrate front-end GUI programs, such as those written with Powersoft's Powerbuilder, to the 4GL Magna code that resides on a server. Magna also provides an API to execute transactions on relational database management systems (RDBMSs). Developers can also use the API provided by their RDBMS vendor to access data stored that database.

Component-based

Software

Procedural Programs. Aside from interfaces, developers can create decomposable applications within a single programming language. This requires up-front planning and discipline. With traditional programming languages, developers must define portions of their code to be segmented into callable procedures. These procedures can be ported to another platform and connected back via an RPC at a later date. Most new mainframe applications are being designed in this way, which makes it easier to migrate these applications to client/server portions. Developers can simply call out the presentation and logic procedures and port them to a client/server platform. Unfortunately, these applications can gradually lose their decomposability as patches are applied. Companies must be diligent about maintaining a strict separation of application components.

Objects Extend

Decomposability

Object-oriented applications extend the decomposability of applications to much finer levels of granularity. Objects can represent presentation, functional logic or data, or smaller elements within each of these components. For example, an object might be a radio button on a GUI or an "account summary" command in a bank teller application. Because objects encapsulate their implementation behind a generic interface, moving them from one platform to another is easier. Swapping out the underlying program that executes the object's behavior or methods is also easier.

Three-Tier Deployment Architecture

Enterprise Applications

Often Require Three

Tiers

Companies have discovered the hard way that client/server applications built on two-tier deployment architectures don't scale well enough to support enterprise applications. By "enterprise," we mean applications that support core operational systems that run the company, such as order entry, inventory, distribution, and finance.

Enterprise Application Characteristics

These bet-your-business applications typically have been run on mainframes and minicomputers

complex, involving considerable up-front analysis and design. Traditionally, completing a program has required 50 to 100 developers working for a year or more. The applications are used by hundreds or even thousands of users spread across large geographic areas. Enterprise applications have requirements for high reliability, availability, portability, scalability and performance.

Early Stages of

Deployment

These enterprise applications requirements are difficult for client/server applications to meet. Many leading-edge users claim that only a three-tier client/server architecture can achieve these requirements. However, we are still at an early juncture in the deployment of three-tier client/server applications. There is little empirical evidence to determine whether three-tier architectures or client/server computing in general are up to the task. The few companies that have deployed three-tier client/server applications have done so in support of decision-support tasks. Most companies are still running operational applications in terminal-host environments, whether the dominant platform is a mainframe, minicomputer, or Unix server.

Three-tier Architectures

Are Useful for

Downsizing

Ironically, many companies view the three-tier architecture as the most way to migrate from terminal-host to client/server computing environments. These companies are moving presentation to intelligent workstations and either porting or rewriting mainframe applications to run on intermediate servers. Many of them plan to eventually migrate data off the mainframe once there are sufficiently high-powered Unix systems and RDBMSs. For now, many are content to use mainframes as giant data servers. Some companies, however, are moving all application components to smaller computing platforms.

The Many Roles that intermediate Servers Play

Add Flexibility in

Application Configuration

A three-tier deployment architecture leverages the components of a decomposed application. Since application elements are independent modules linked via some sort of interface, this gives developers or architects the freedom to deploy these components on optimal platforms.

For example, NASA may have an application that helps plot voyages of the space shuttle. Part of that application may involve computing the trajectory of the spacecraft at liftoff to obtain optimal orbit during a week-long voyage. This compute function might perform better if it were calculated on a high-performance workstation or supercomputer. A three-tier application would allow that function to run on a separate intermediate server. Moreover, it would make the function available to current and future applications that may need to perform similar calculations.

A three-tier architecture allows users to create a middle tier of intermediate servers whose purpose

DOCKET ALARM

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

**LAW FIRMS**
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

**FINANCIAL INSTITUTIONS**
Litigation and bankruptcy checks for companies and debtors.

**E-DISCOVERY AND LEGAL VENDORS**
Sync your system to PACER to automate legal marketing.