A GENERALIZED WAY OF THINKING ABOUT N-TIER CLIENT/SERVER ARCHITECTURES

by George Schussel

The classical client/server computing model (illustrated in Figure 1) is evolving as companies gain experience with it. Client/server computing emerged as a solution to performance problems with file server systems as they began to scale upwards from a few users to a few dozen. The file server approach (popularized by dBASE and FoxPro) is to transfer files from a shared server to the desktop PC. That approach is simple and works as long as shared usage is low and the volume of data to be transfered is low compared with LAN capacity. As customers demanded GUI computing against shared databases, demand for client/servers grew rapidly over the last few years. By now, the idea of using Windows or Mac style PC's to front end a shared database server is a familiar and widely implemented.

The 2-tiered client/server architecture has proven to be very effective in solving workgroup classes of problems. "Workgroup" is loosely defined as six to 50 people interacting on a LAN. For bigger, enterprise class problems and/or applications that are distributed over a WAN, use of this 2-tier approach has generated some problems. What typically happens with a 2-tier architecture is that performance deteriorates as the scale increases. One corresponding result is that this architecture is expensive to implement over a WAN.

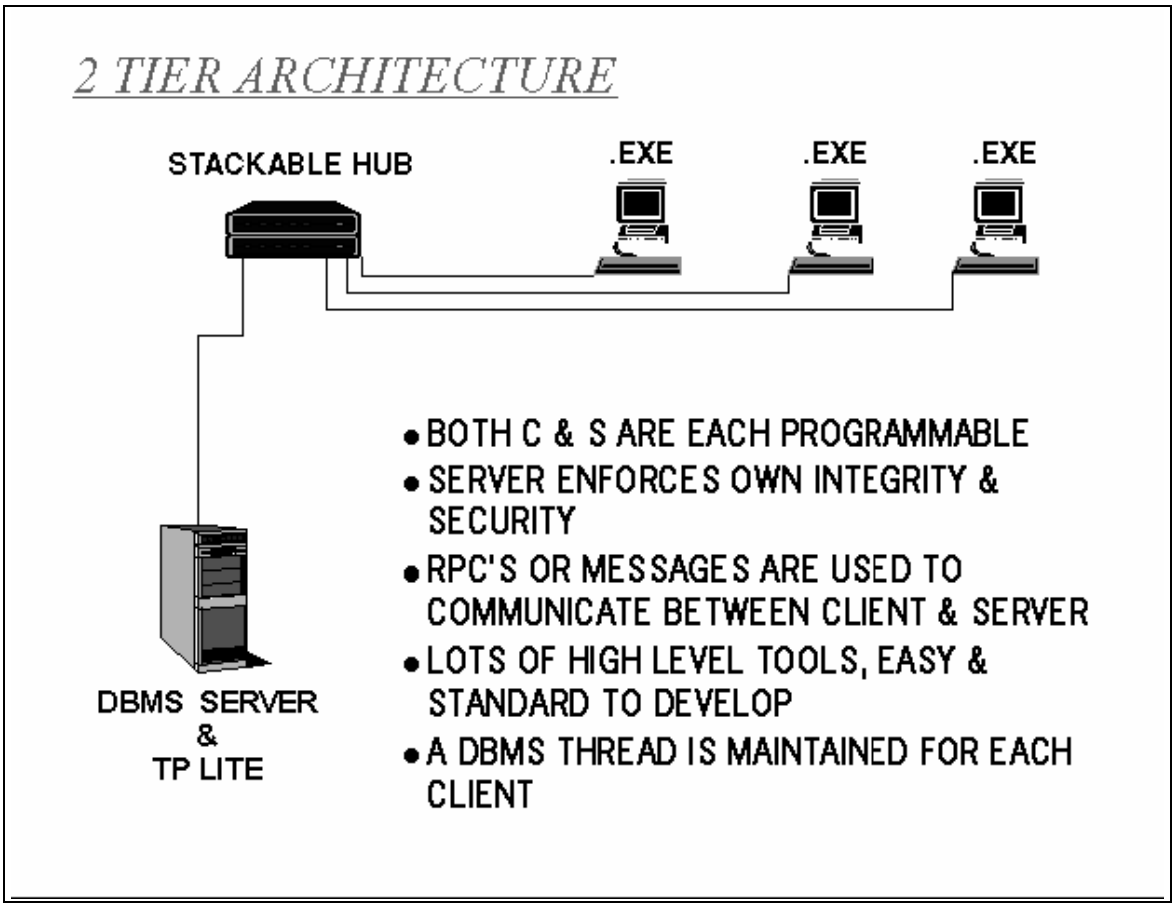

FIGURE 1 - THE BASIC 2-TIER ARCHITECTURE

The software tools community and advanced users have responded by combining some new and some old ideas in new forms creating a “3-tiered” architecture for client/server. The advantage of 3-tier is that it can be used for larger and more widely distributed applications. This 3-tier architecture (illustrated in Figure 2) looks much more traditional than a 2-tier approach because most of the application’s logic has been moved off the PC and into a common, shared server. The PC is basically used for presentation services - not unlike the role that a terminal plays. Since the application server doesn’t need to worry about driving a GUI, it can devote all of its resources to business logic. As a server it can run an efficient multitasking OS like NT, OS/2, or UNIX. These server OS’ also run on symmetric multiprocessing (SMP) configurations, therefore, the server is much more scaleable for performance than a PC. In addition, as new versions of the application software are developed and released, the installation of that software occurs on the single server rather than hundreds or thousands of PC’s.

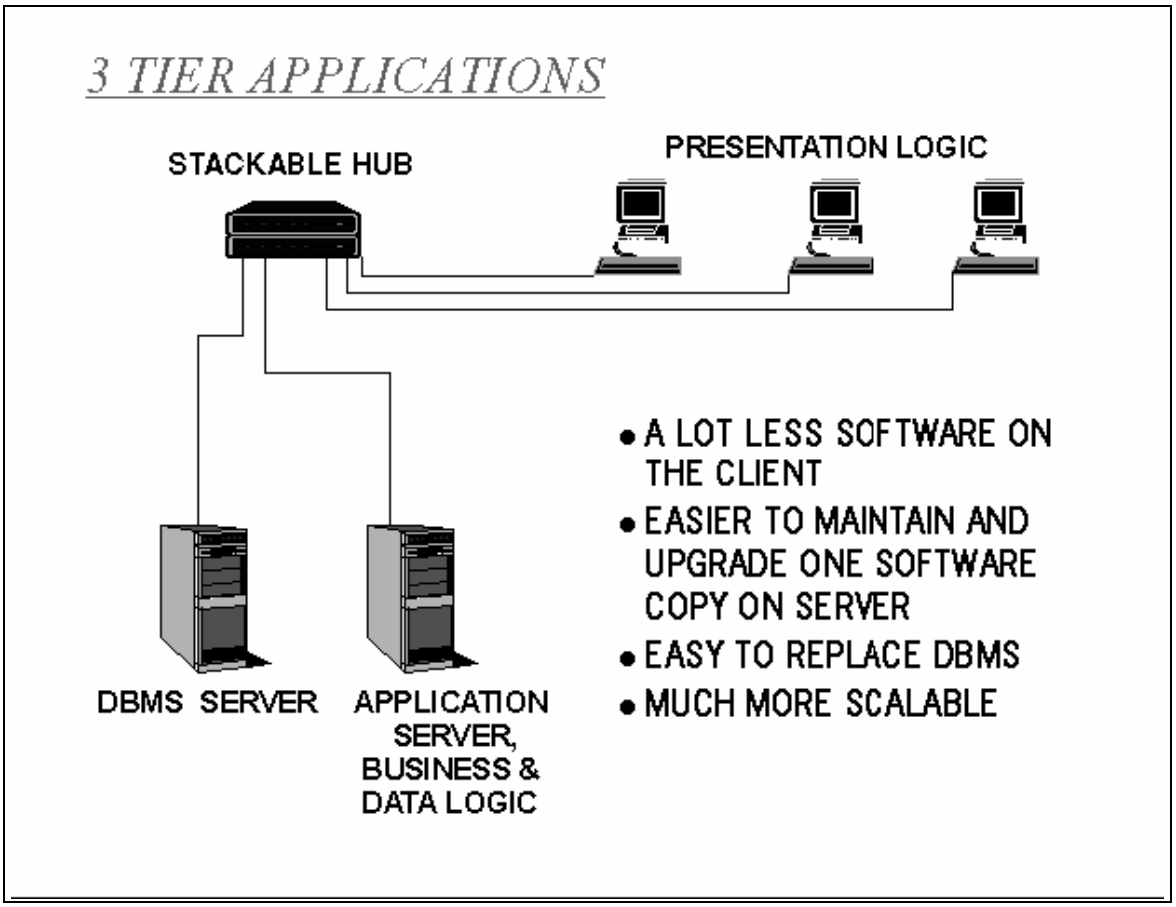

FIGURE 2 - A 3-TIER APPLICATION SERVER ARCHITECTURE

One particular type of application server is known as a transaction processing or TP monitor. Almost all interactive applications in the 1970's and 1980's were implemented with TP monitors like IBM's CICS. TP monitors haven't been used in 2-tier client/server architectures. That's because many of the services provided by a TP monitor have been included as part of the DBMS services provided by vendors like Sybase and Oracle. Those embedded (in the DBMS) TP services have acquired the nickname "TP Lite".

It's most useful to think of a TP monitor as a kind of messaging service. It allows applications on a PC to asynchronously connect to the TP monitor, which acts as a queue server. The transaction is accepted by the monitor, which then takes responsibility for managing it to correct completion. For a really scaleable application, a true TP monitor is an absolute requirement. Some other key services that a monitor provides are:

- The ability to update multiple different DBMS in a single transaction
- A facility for developers to ignore different OS and DBMS engines and just write to the TP monitor interface
- The ability to attach priorities to transactions
- Support for data stored in non-RDBMS DBMS' and files

- Robust security, including Kerberos

With all those positives it might seem that all client/server applications should be written with a TP monitor; however, there is a downside to the technology. That downside is the added complexity of the application code - and the necessity of writing that code, as opposed to using visual generators like PowerBuilder or Visual Basic.

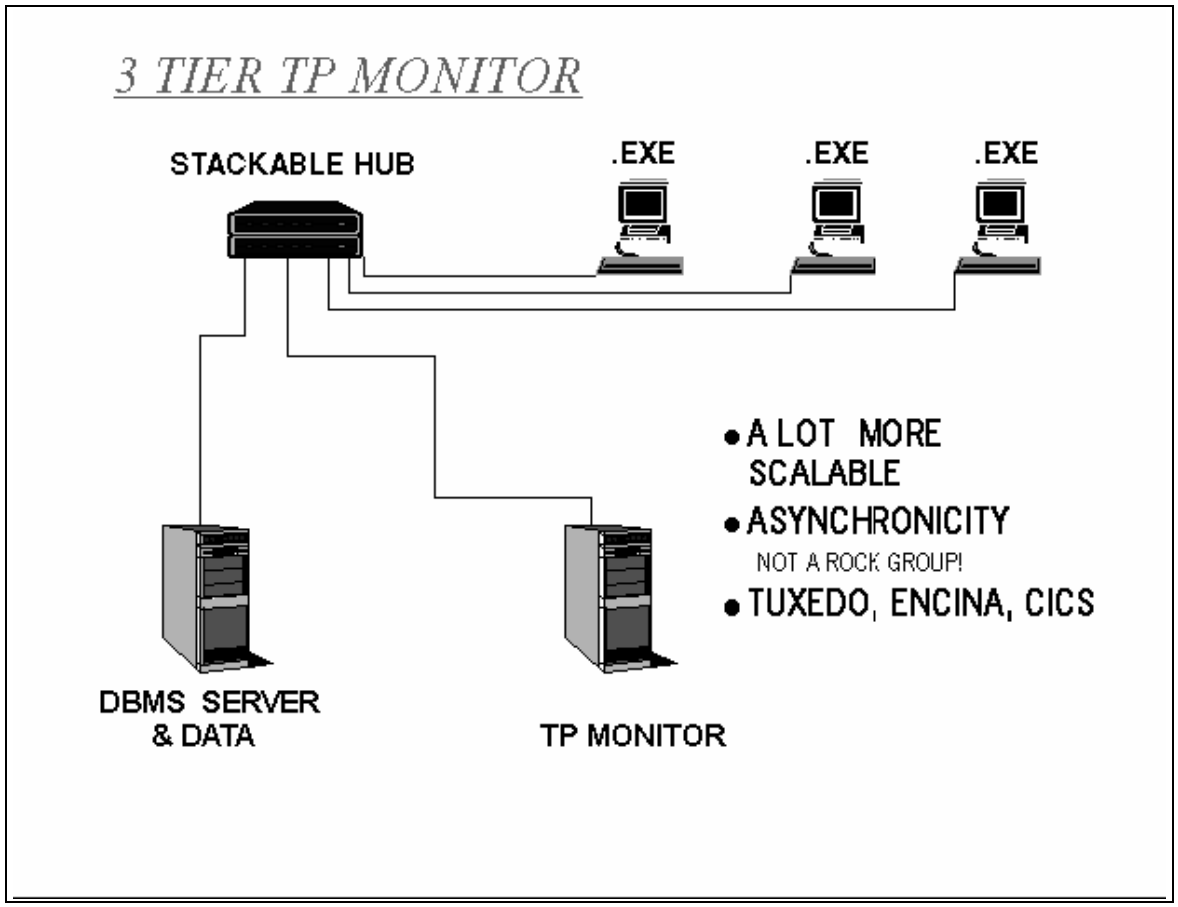

FIGURE 3 - A 3-TIER ARCHITECTURE WITH TP MONITOR

A 3-tier architecture is also useful for data mining or warehouse types of applications. These applications are characterized by unanticipated browsing of historical data. The databases supporting this type of application can sometimes be huge (up to a few terabytes -10^{12} bytes); therefore, the structure of the database and the tools used to browse it must be right or the performance becomes unacceptable.

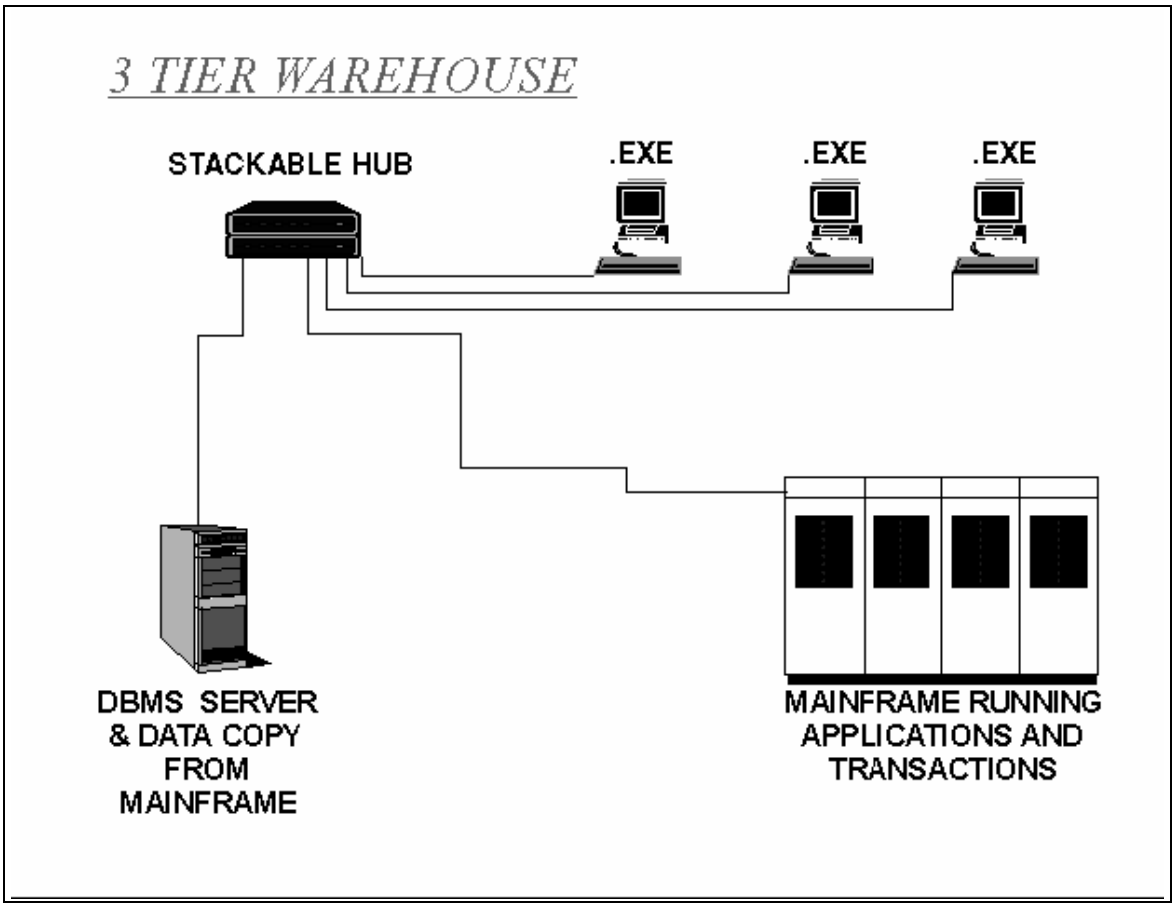

FIGURE 4 - A 3-TIER ARCHITECTURE FOR DATA WAREHOUSING

Because long, time-consuming database browses can negatively affect the performance of the on-line systems that update data, the best and most typical way of creating a warehouse application is to create a data copy of the necessary information and load that data copy on its own server (Illustrated in Figure 4). Regardless of the form of storage of data on the mainframe TP application(s), the data copy is usually stored in the DBMS server in a relational DBMS or an OLAP (on-line analytical process) server. When the data copy is created, value is usually added in the form of denormalization and summarization.

Three different types of 3-tier architectures are introduced in Figures 2 through 4. Since it's still early in the technical evolution of client/server computing, it's very unlikely that these three different examples of new architectures are the only manifestation of multi-tier client/server architectures. In fact, it's easy to imagine the combination of two or more of these approaches into a "4-tier" architecture as illustrated in Figure 5.

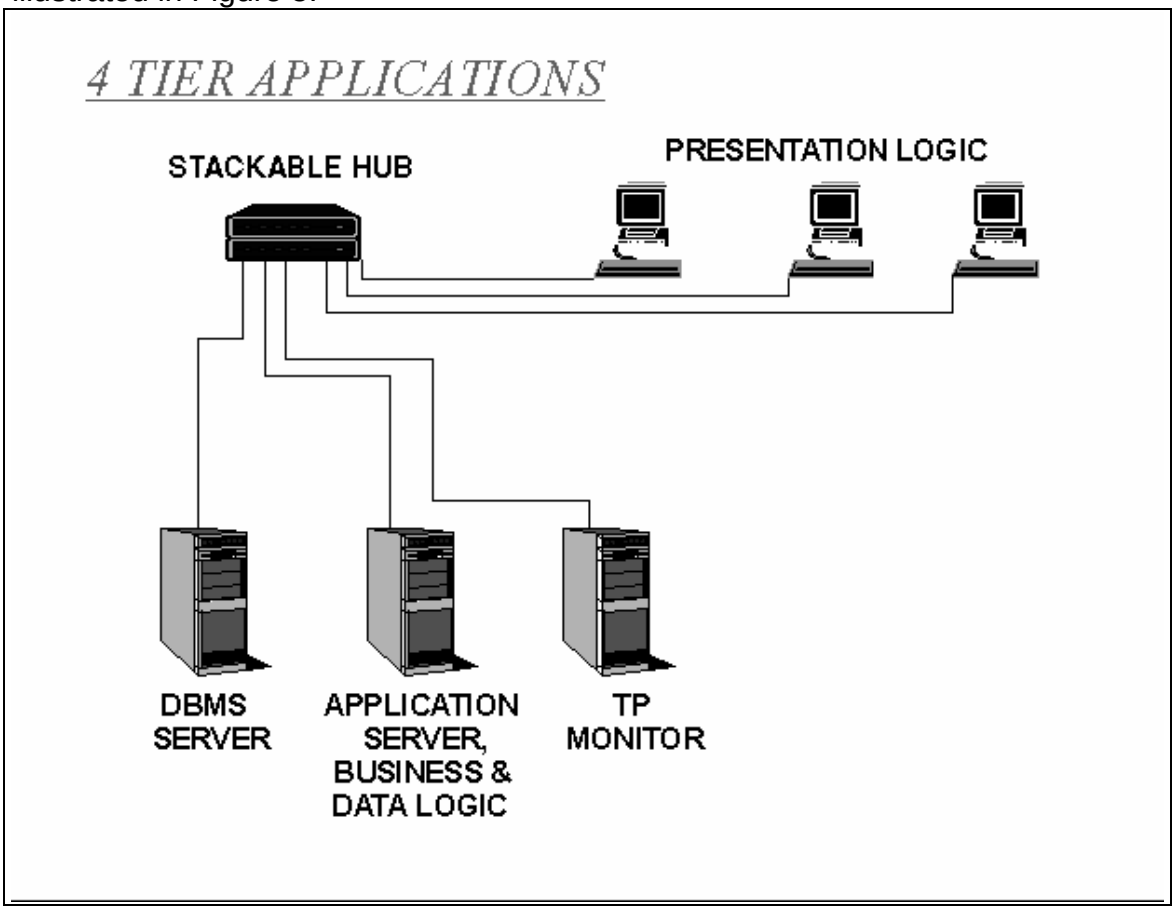

FIGURE 5 - A 4-TIER ARCHITECTURE WITH APPLICATION SERVER AND TP MONITOR

Here we're using an application server to centralize most of the business logic and a TP monitor to provide the advantages listed above. While certainly more complicated to develop than traditional 2-tier approaches, this type of application architecture is very scaleable and would be an appropriate structure for developing enterprise wide client/server applications.

As the application building tools, systems management, and DBMS products for client/server become more mature, we may see companies migrating to mature general purpose “N-tier” architectures. Such an approach, which combines all of the concepts mentioned above is illustrated in Figure 6.

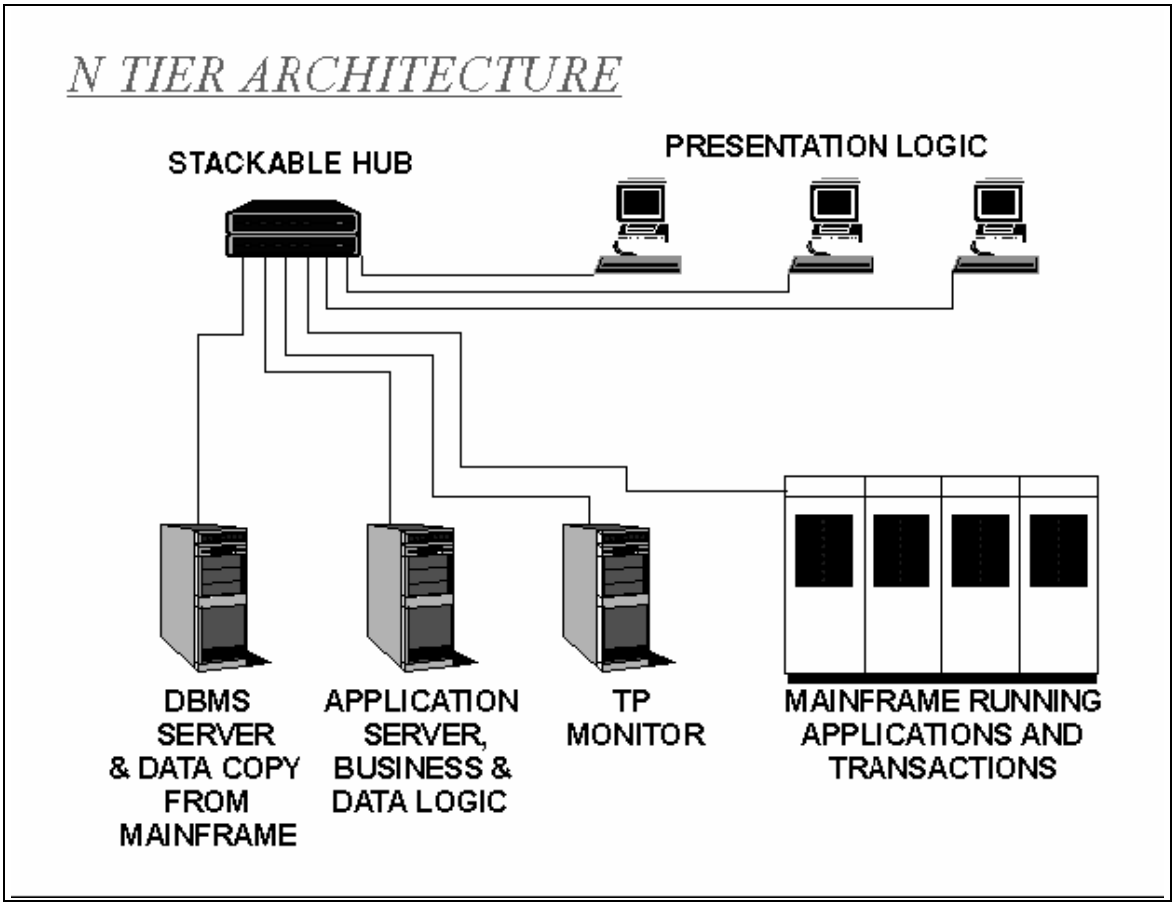

FIGURE 6 - AN N-TIER ARCHITECTURE CUSTOMIZABLE TO NEEDS

The diversity of these approaches confirms the power of the client/server architecture. It allows the designer to mix and match technologies in order to meet the needs of the environment.

Schussel is the founder and Chairman of Digital Consulting, Inc. (DCI) in Andover, MA and Chairman of the Database & Client/Server World trade show. Reach him at 74407.2472@compuserve.com or http://www.dciexpo.com/.