NOVEMBER 1991

A Comparison of Mainframe and LAN Database Process Architectures

Rich Lee
Senior Consultant
Systems Research Department

This Application Note depicts the major database process architectures used in both mainframe and LAN environments. Each architecture is reviewed with an eye to migration applicability, which should provide insight for the LAN database administrator or developer. This AppNote is one in a series on LAN database implementation strategies.

Contents

Mainframe Database Architectures

Database technology in the mainframe/minicomputer environment has evolved steadily over the past 40 years. In the course of this evolution, numerous models have emerged for designing database configurations and structures. Process architectures such as host-to-terminal, host-to-host, cooperative processing, and distributed database were developed to meet the changing needs of the information processing world.

With the introduction of LAN technology, the database community began to look at ways of incorporating these mainframe database configurations and structures into LAN-based models. There are, after all, certain similarities: host-to-host looks a lot like peer-to-peer; cooperative processing appears related to distributed processing; and distributing the database itself seems a lot more flexible on a LAN.

However, the migration of mainframe process architectures to LANs is not as easy as it might first appear. Perhaps part of the problem is that the 40-year evolution of mainframe technology was compressed into six or seven years of LAN evolution. With technical improvements occurring at that rate, it's difficult to clearly discern changes in database structure and design. Just keeping up with the technology and making sure we all use the same terms is a big challenge.

To further complicate things, no one architecture seems to work best for every situation. Some designs call for two or more architectures to be applied simultaneously, thus increasing the number of variables to contend with. As the problem-solution index grows, the solvability index decreases. Without understanding the processes involved, debugging LAN database designs becomes a living nightmare.

This AppNote presents some of the basic architectural issues involved in migrating databases to LANs. It clarifies some of the semantic confusion to form a common foundation on which to build future AppNote discussions. It then depicts several of the mainframe process architectures that can be applied to LAN-based systems.

Migrating to LAN Environments

Prior to the emergence of the LAN marketplace, many mainframe database concepts were already being migrated directly to the PC, some in part, some in whole—a top-down migration.

Within a few years after the introduction of LAN technology, PC database applications and related programs were rewritten to take advantage of the LAN's file sharing capabilities—a bottom-up migration.

In these migration strategies, database designers foresaw a whole new development environment and new opportunities for their 3GL, 4GL, and object oriented programming products. Of greater interest to the database community were the translations of host-architected database process architectures to the LAN. For example, client-server technology (originally developed for the mainframe) found another real existence on the LAN. Network managers, however, began to face difficulties like those encountered by mainframe designs in previous years. Not the least of these was communications between network servers, like host-to-host communications.

Semantic Difficulties

One of the roadblocks to solving these difficulties is the lack of common understanding between the parties involved. You have probably noticed that highly-trained professionals tend to argue the pros and cons of their particular areas of expertise whenever they meet. For instance, mainframe experts might argue the merits of downsizing scenarios with LAN experts, especially in regards to communications or database issues. Semantics, lack of clarity, and general misunderstandings can create a lot of heat in these discussions.

For example, within the LAN environment the term "distributed processing" is often used. At Novell, we use the term to describe the classic LAN environment in which processing power is distributed among all the intelligent nodes on the network, as in Figure 1.

Figure 1: Novell uses the term "distributed processing" to describe the classic LAN environment.

However, this distributed *processing* environment on the LAN is much different from the distributed *database* environment discussed later in this AppNote.

There are very different approaches to information processing, with unique concepts and solutions that need to be appreciated by all parties. While many of the labels might be different, the underlying process architectures remain the same. This AppNote's approach to the subject will explore the latest database buzzwords and hopefully clarify their meanings.

Process Architectures

What is a process architecture? Essentially, it is a model that describes the component parts of a process. In our case, the "process" in question is the process of sending data to or retrieving data from a database.

I've chosen a diagrammatic approach to depict the various database process architectures, along with a layout of the hardware. Hopefully, this manner of presentation will help clarify some of the conflicting terminology surrounding database implementations on LANs.

Understanding the Diagrams

Each of the following diagrams has two sections. The first section depicts the process architecture. A two-level box represents the component parts of each host, terminal, server, or workstation. The label "Application" indicates the location of the application; "DBMS" indicates the location of the database management system. Where a component is shown with dotted lines, it means that the component does not exist in that particular node.

The second section of each figure presents a high-level hardware diagram for structural considerations. Again, the terms host, terminal, server, and workstation are used for clarity.

Certainly these diagrams are not conclusive; they merely describe the ways that a particular process might be implemented. But they are operating system independent. For instance, distributed processing could refer equally to distributed processing under NetWare Lite, OS/2, or UNIX.

Whereas one picture is worth a thousand words, two or three perspectives of the same drawing can become confusing. Unfortunately, for some of the definitions, the picture has changed over the years. Now, there may be more than one hardware structure for any one process architecture. Specifically, IBM's view of peer-to-peer and client-server architectures has changed significantly from the original conception.

In another AppNote, I will relate the process architectures to the more prevalent LAN architectures, and then show some of the actual software installations and discuss operational constraints.

On-Line Terminal Entry

One of the earliest process structures is on-line terminal entry, as illustrated in Figure 2. This architecture does not have a LAN equivalent.

Figure 2: In the on-line terminal entry process architecture, the host does all the processing.

Note that in the on-line terminal model, neither the application nor the DBMS runs at the terminal. This is strictly a mainframe/minicomputer approach to data entry and queries. The central processor does just that —"centrally process" everything!

Peer-to-Peer

On the other hand, peer-to-peer is a mainframe process architecture that has worked its way to the LAN. Figure 3 depicts the peer-to-peer model as originally defined by IBM.

IBM originally proposed the peer-to-peer architecture for database communications between host systems (host-to-host). In the LAN environment, the "peers" can be servers or workstations. Thus you can have a server-to-server, workstation-to-workstation, or even server-to-workstation interaction, as shown in Figure 4.

One aspect of the peer-to-peer architecture is that an application is loaded at each client, and that application expects to communicate with other applications as servers or clients. However, with certain operating systems (such as NetWare Lite) any client can also function as a server.

Figure 3: In the original peer-to-peer architecture, two like machines pass data back and forth.

Figure 4: The concept of peer-to-peer has been expanded to include unlike machines acting as peers.

Cooperative Processing

Another of the process architectures, cooperative processing, is shown in Figure 5. Note the use of IBM's HLLAPI in this architecture.

Figure 5: In cooperative processing, the total task is divided up between processors.

"Cooperative processing" is one of the more confusing terms, because it is used to denote almost everything. It can mean that more than one processor might possibly do work on application data simultaneously. It can also mean that a client and a server exist, each processing some part of the data record for the DBMS.

I have heard two different explanations of cooperative processing on the LAN: client-server and file server-based. Neither was particularly accurate from my perspective, because cooperative processing has to do with application (procedurally) oriented services, not I/O or DBMS.

Client-server models have been touted as cooperative processing. While there is a lot going on at the client and at the server from the data's point of view, this is not cooperative processing. It is client-server processing. When you move away from application processing to the DBMS, the cooperative side of the process is over with. DBMS processing is (hopefully) rule-based, data-bound, and not particularly cooperative with anything except the DBMS. This applies particularly to SQL implementations that employ the relational model.

In this and future AppNotes, "cooperative processing" means using other processors for cooperative application-based processing, as shown in Figure 6. We'll reserve the term "client-server" specifically for DBMS-based processing.

Figure 6: In the LAN environment, cooperative processing occurs at the application level.

Here is an example of LAN-based cooperative processing that may be useful. Picture a scenario in which large, ASCII payroll files are presented to the Human Resources department for incorporation with the employee records. The ASCII files are 10MB in size, and three such files come in every month. A single 386/16MHz machine could take up to eleven hours to parse, convert, and write the 10MB file to the employee records (file server-based single CPU). If multiple machines were employed and each accessed different parts of the 10MB file, imagine how quickly two or three workstations could accomplish this task!

Client-Server

Client-server technology is a strong process architecture well suited to database management systems. At the same time, it is one that is often misunderstood in LAN-based database management systems.

When client-server architecture is discussed with regard to a DBMS, the server is usually a strict data server, as shown in Figure 7.

Figure 7: In the client-server database architecture, the client application makes requests of the server DBMS.

In the client-server process architecture, there is more than just a server DBMS responding to the application requests of the client. One of the intrinsic properties of the client-server process architecture is that the client sends the server only one request for information at a time, as shown in Figure 8.

This differs from the case of an application dealing with a file server where a request might be an application program parsing through the entire contents of the database. In client-server technology, database methods are run at the server component (procedural implementations) or at the DBMS (declarative implementations), not at the client component.

Figure 8: Characteristically, the client sends the server only one request at a time in a client-server relationship.

The picture can get even more confusing when you start to combine architectures. For example, if you are working in a peer-to-peer hardware architecture such as NetWare Lite, where each station on the LAN has the ability to work with a server (client) or to share resources (disks, directories, printers) with other clients (including a server), you have a client-server process architecture running under a peer-to-peer architecture.

Distributed Database

Distributed database is a term often used in the LAN and database communities. It is unfortunate that the words "distributed database" are used with several connotations in the computer industry. In some usages, the speaker might mean that the data is distributed by the server (host) to the workstations; in others, that the databases (files) are located on different servers in the same network. Both of these usages are correct with regard to distributed data, but incorrect with regard to distributed database management systems.

The correct DBMS usage is that the data resides at data servers or DMBSes in different LANs on the network *or* at different geographical sites. Figure 9 illustrates various distributed database possibilities.

Figure 9: In a distributed database architecture, the DBMS resides at various locations on the network.

While many people use the words "distributed" and "database" together in the same sentence, it is important to make sure that what you are talking about really corresponds to the other person's understanding. By definition, a distributed DBMS or remote distributed DBMS is a collection of separate or geographically independent DBMSes that have at least one common application.

In some cases, the data is termed as "distributed" because it is distributed across several floors (different LANs) in the enterprise. In other scenarios, the data might be distributed across the continental United States and would be termed "remote distributed data."

Unfortunately, it is difficult to tell if distributed data is truly a distributed database, for it would be equally possible that each of the databases shown could be local databases without providing data access to a common application.

Conclusion

In reviewing the different process architectures in this AppNote, I have tried to provide a conceptual basis for discussing LAN database implementations. In future AppNotes, I'll discuss the hardware factors and actual LAN integration procedures.