In this installment of the Tech Resource Zone, BASIS Chief Technical Officer John Schroeder finishes his two-part series on BBj. Here, he explores remote processing considerations with BBj.

Client/Server Computing: Comparing PRO/5 and BBj

by John Schroeder

In the last issue, I wrote about the various editions of BBj, and the installation and licensing procedures. Here, I will focus on networked systems and the differences between PRO/5® and BBj® Enterprise Edition.

Networks come in several different flavors, from local area networks (LANs), in which the nodes and the network connections are in the same location, to wide area networks (WANs), in which at least some of the nodes are remote from the main server. LANs and WANs are fairly tightly coupled, private systems, normally serving one business or organization. The Internet is a loosely coupled system serving as a medium for widely separated clients and servers operating on a public network. All network configurations are similar in that they involve servers providing their clients with applications and data.

In PRO/5, a client/server system is built around the PRO/5 Data Server®. PRO/5, Visual PRO/5®, and ODBC clients connect to the PRO/5 Data Server in order to access data on the system server.

In BBj, not surprisingly, the client/server system is also built around a data server, in this case, the BBj Data Server™. The comparison, however, stops there. There are vast differences in the architecture of the BBj Data Server that make it inherently better both in multiuser and in client/server configurations. In addition, because the major BBj components are Java based, the availability of Java Runtime Environments (JREs) makes them platform independent. Thus, the various components of a networked BBj configuration can run across multiple operating system (OS) platforms.

PRO/5 Architecture: Thick Client

In PRO/5, the client runs the application and relies on the PRO/5 Data Server for the data needed by the application. A fully functional PRO/5, Visual PRO/5 or ODBC client module runs on the client machine, while the remote file system, provided by the data server, runs on the server. This is a so-called "thick client" configuration, where the client has the full complement of software and handles all of the processing, while the server handles only data requests. This configuration has the advantage of distributing the required application processing among all the machines in the network. The disadvantage is that data access across the network can be tens or even hundreds of times slower than local data access. Applications that process large amounts of data do not run efficiently in this configuration.

Think about the situation involving an update program. Typically, these programs read data from several files, process it and then write data out to multiple files. In a thick

client configuration, all these data travel back and forth across the network. If data access across the network is 50 times slower than a local access, then the total transfer is about 100 times slower. The more data, the slower the application runs. An update that performs very well when running locally on the server can slow to a crawl on a network. A design that allows the data to be processed on the server is much better suited in this case.

BBj Architecture: Thin Client

Enter the BBj Data Server, which forms the hub for all BBj components. Several architectural features make the BBj Data Server inherently more robust than its PRO/5 counterpart. One of the main features is that all data requests, whether from local jobs, remote BBj clients, ODBC or JDBC clients, as well as special purpose applications written in Java or C++, are processed in the same file system. This allows the BBj Data Server to manage caching of data, operating system calls, and record and file locking in a much more efficient manner. This benefit applies whether the client is local or remote.

In a BBj network configuration, the network administrator has a choice as to where the application will run. It can be configured to run on the client, making data requests across the network as in PRO/5. Or it can be configured to run on the server, where data requests are handled locally and only screen update information is sent across the network. The benefit to running in this "thin client" configuration is enormous when the application requires processing of large amounts of data.

Remote Processing With BBj AppServer

The bridge between the remote client and the BBj Data Server is the BBj AppServer™. This is basically the BBj Interpreter with a socket manager providing a connection to the client. The client sends a request to the AppServer to run a program. The AppServer starts up an interpreter thread and manages the flow of information between the BBj Interpreter™ and the client. The interpreter interfaces with the BBj Data Server to handle the data requirements of the application. The AppServer/Interpreter and BBj Data Server run together as the BBjServices application in one JRE. With the integrated BBjServices running, all communication between the components is direct with no network intervention. Even tighter integration is possible on a single system in which the AppServer interface is bypassed and the Interpreter communicates directly to the client and the BBj Data Server without the intervening layer of a socket connection.

The thin client environment is similar to a multiuser system in the PRO/5 world, where terminals connect the user to PRO/5, which includes the interpreter and file system in one package. Both configurations provide high-speed data access for the application. The advantage of the BBj configuration is that the thin client can handle graphical user interfaces as well as character displays and can be located anywhere there is either a public or private network available. Last, but not at all least, because the thin client is a Java component, it can run in a JRE that is plugged in to a Web browser.

So, BBj applications can run very efficiently across a wide spectrum of environments, from a multiuser terminal configuration to a widely distributed network using browsers and any combination in between.