Flexible File Management

by Dale Frederick

File access in BASIS' new product generation, BBj™, will be structured differently than it has been in the past. All access from within BBj programs will still operate on a channel. But within BBj, the internal implementation will be a dramatic departure from what has existed in previous generations of the BBx®language. In this TechCon99 session, I'll be explaining and showing you these differences.

Channels that operate on devices will be implemented locally within BBj, as will access to STRING files. But on-disk structured files, such as the MKEYED file type, will be implemented and accessed through a new BASIS Database Server™. The new database server fully supports all of the existing BASIS structured on-disk file types and access methods without any changes to existing BBx code.

Enhancing Performance

The BASIS Database Server will be fully implemented in the Java™ programming language and operate as a stand-alone and self-contained server that executes within a single Java Virtual Machine (JVM) instance. The database server consists of several independent but cooperative subsystems. Each subsystem is designed to handle a specific task related to the manipulation and/or management of structured on-disk files. Some of the subsystems resemble the functionality found in many modern multitasking operating systems and rely on Java's threading support as provided by the Java programming language, class libraries and the JVM for multithreaded operation. Many of these subsystems will live as daemon threads exclusively within the server.

One of the goals of the new design and implementation of the structured on-disk files is to improve the performance of file operations. Many of the performance bottlenecks that occur with PRO/5® are caused by the extensive use PRO/5 makes of operating system services. PRO/5, to aid in the arbitration of shared access to files, makes many system calls to the operating system to perform operations such as:

- File locking: Segment locking is the mechanism used by PRO/5 to arbitrate access to file headers, key chains and records. To communicate these actions to other users of a file, operating-system locking calls are used. The operating systems themselves maintain an internal lock database, relieving PRO/5 from having to manage lock data and from having to do deadlock detection.

- Setting and clearing signals and timers for I/O operations: Many I/O operations are timed automatically by PRO/5. The OPEN verb, for instance, utilizes system timers to interrupt an operation that may be blocked because a resource is locked or unavailable. The programmer may also utilize a timer (the **`TIM=`** option available on some verbs) for some operations.

- Reading and writing various structures within a file: there is obviously no way to get around the absolute need to physically read and write the contents of a file.

The various operations that modify a file must occur in an orderly and controlled fashion to preserve the integrity and structure of the file. Modern operating systems such as UNIX and

Microsoft Windows (NT, 95, 98) present an environment where multiple users can read and modify a file in an unpredictable and random manner, when viewed against the activity of other users of the file.

As a general rule, however, the system calls used in the operations are both time consuming and, in some cases, expensive. By internalizing into a single server much of the activity for which PRO/5 relies on an operating system, the number of system-calls can be dramatically reduced. That is one of the design goals of the BASIS Database Server: to reduce the system call overhead associated with the management of BBx structured files. Other advancements and data management techniques will be employed to manage files, some of these techniques are discussed here. I'll elaborate more on these and others in the session.

BBj File Services

In BBj, all of the BBx verbs, such as OPEN, READ, WRITE, will map to service requests and messages between BBj and the database server. Although this sounds a lot like the product known today as the PRO/5 Data Server®, it is dramatically different. Whether one user or 1,000 users have a file open, only a single database server will be executing. Additionally, only a single instance of the physical file will be opened and maintained by the server. Many of the operations that are handled though system calls will be reduced or eliminated by utilizing new techniques in a multithreaded environment for the open files, regardless of the number of external users that have the file open.

With multithreading, multiple user requests could be simultaneously searching the same key chain of a MKEYED file. This of course assumes that the executing JVM actually has the ability to execute multiple threads simultaneously on a multiprocessor machine. To do this with PRO/5 today between two or more users could require extensive process-context switching by the operating system.

SQL Services

Allowing the SQL engine to run as another service of the database server, much like the BBj file services, will allow the SQL engine to provide SQL services while providing the same level of access and sharing of resources. Both a JDBC™ driver and a new BASIS ODBC Driver® will have the ability to fully utilize the remote execution of SQL services.

Locking

The tracking of locks will be done internally to the server. In many cases, the locks that are utilized in PRO/5 are used to synchronize activities between users of a file. By utilizing advanced internal synchronization techniques within the server, a large amount of locking overhead will be totally eliminated. For locks that actually implement the locking of records (EXTRACT), an internal lock database will be utilized.

Caching

I mentioned previously that there is no way to get around the need to physically read and write the file. That doesn't change much with the incorporation of file operations into a server. However, having a single place for all users to request file services will allow for a global cache of various segments from the file. This will allow for increased speed in the reading of a file and in the searching of various keys structures when both reading and writing a file. As the server develops over time, tunable parameters will be added that should allow the sizing of caches by file and possibly by key.

Additional Services

The ability for Java programmers to access BBx files directly will also be provided by a package of classes that BASIS will provide. Additionally, a C-interface library is planned that will allow C programmers complete access to the BBj file services.

Dale Frederick brings more than 15 years of software development experience to BASIS as a Senior Software Engineer. Before joining BASIS in 1995, Dale was with Digital Equipment Corporation. He has extensive knowledge of UNIX, VAX/VMS, and clustering internals, as well as TCP/IP and DECnet networking. Dale is the technical lead for file system development in BASIS products.