



**A Practical Look at Benchmark Results:
tableBASE Boosts DB2 Throughput**

By Armin Kastner

Background

DB2 remains the database management system of choice for large-scale database processing on the mainframe. For many organizations, the need for greater throughput on their mainframe applications is increasing, and will continue to do so for several years to come.

Based on benchmarks conducted by DataKinetics® and IBM® in March 2004, tableBASE® has been shown to be between 19 and 23 times faster than DB2 for certain types of table processing. These benchmarks, the details of which are available from DataKinetics, show that a batch architecture using tableBASE, in conjunction with DB2, can substantially reduce CPU overhead. Although the benchmarking was limited to batch applications, the performance of on-line applications can also be significantly improved through the use of tableBASE as well.

tableBASE tables reside in tableSPACE regions (TSR), and each region can consist of up to 2 Gigabytes of memory. Another DataKinetics product, Virtual Table Share (VTS), supports the sharing of the data in tableBASE between different batch programs, DB2 Stored Procedures, CICS and IMS. On-line transactions from CICS or IMS can access large memory resident tables with very little overhead, and possibly avoiding most, if not all, access to the database. J2EE or .NET framework transactions, making use of DB2 Stored Procedure calls, will also benefit from the performance benefits of tableBASE by placing read only tables into a DB2 Stored Procedure shared tableSPACE region.

The architecture described in this report is based on the use of VTS-TSRs, which is the shared tableSPACE region.

Overview

This report will highlight the significance of the tableBASE benchmark results and illustrate how tableBASE can be used to increase DB2 throughput using a large-scale tax return processing batch architecture as an example application. Large scale tax return processing is representative of a class of batch processes where the batch file to be processed can be partitioned into a number of smaller batch files. These smaller batch files can then be processed in parallel, using multiple currently executing instances of the same program, in order to make maximum use of available mainframe hardware resources. Using tableBASE, these programs can then share a single copy of read only tables at memory speed.

Any batch program that needs to access large amounts of read data can benefit from tableBASE. Tax return processing was selected as an example because of the on-going pressure in most tax departments to process more tax returns in a smaller period of time and tableBASE is an excellent way to achieve greater batch processing throughput.

Batch Architecture Example

tableBASE Copybooks and Reference Tables

Physical database design for high-volume batch processing may consist of partitioning key database tables/tableSPACES based on key ranges, or some other technique such as hash key(s), which will support application controlled parallel processing. Parallel processing will then consist of running multiple instances of the same job, in parallel, with one job per partition. The amount of parallel processing will depend on the hardware configuration.

Using tax return processing as an example, each of these jobs will have a large number of reference tables in working storage to support edits based on business rules. These reference tables are usually maintained in copybooks. tableBASE can eliminate the use of these copybooks and therefore help reduce maintenance costs. Instead of copybooks, the reference tables can be loaded into a tableBASE VTS shared data space. The resulting memory savings by having a single copy of the reference tables in memory instead of one copy for each parallel job instance will depend on how many instances of the job are running in parallel and the size of the reference tables involved. The increased path length to process the reference tables in tableBASE as opposed to working storage is more than compensated by elimination of recompiles and the savings in memory.

tableBASE can eliminate recompiles due to copybook reference table changes as well as significantly reduce memory usage when running multiple instances of the same job in parallel.

tableBASE and Parallel Processing

Using the tax return processing example again, a batch file of tax returns, sorted by an external key, is processed against DB2 database tables that have clustering indexes based on the same external key. This external key may consist of the Social Security Number, for example. The clustering index may result in DB2 using dynamic pre-fetch to access the taxpayer information using the external key. The taxpayer information retrieved (including the internal key, used for partitioning) may be needed again for the parallel processing of the actual tax return, and tableBASE is an excellent way of retaining specific taxpayer information in memory in order to reduce the amount of database access during parallel processing

It should be noted that in the March 2004 benchmark that DataKinetics performed with IBM, one test consisted of the equivalent of a 'SELECT... WHERE...' statement for 1,000,000 rows. The unique key consisted of 20 bytes. In this test, the CPU ratio between tableBASE and DB2 version 7 was 1/22 and the ratio to DB2 version 8 was 1/23. The DB2 databases were fully tuned by IBM and many IBM customers might obtain even better results depending on the configuration.

Not only can the tableBASE table be accessed much faster than if the table were in DB2, but all tableBASE tables can also be shared by multiple programs and DB2.

In the case of the tax returns processing example, as soon as the single tax returns table has been split into separate files for each of the parallel jobs and sorted by the internal key (the same key used for range or hash partitioning and clustering index), parallel processing can start. The advantage of having the taxpayer information required for further tax return processing in tableBASE tables is significant because taxpayer data required for the tax returns processing is available at least 22 times faster than accessing it in the database, thanks to tableBASE.

'Heap' Table Processing

A 'heap' table is a table that has no indexes (and therefore no primary key). This type of table can be used to replace a flat file for DB2 batch jobs with the advantage that DB2 will roll this table back to the last commit point in case the batch job writing to this table fails.

Depending on the batch commit frequency, tableBASE can provide a high-speed, low overhead alternative to 'heap' tables, if the tableBASE table is stored to disk when a commit is performed. The time required to write the tableBASE table to disk must be judged against the DB2 overhead of using a 'heap' table.

The DataKinetics/IBM benchmark showed 414,938 rows could be inserted into a tableBASE table per CPU second, as opposed to 21,935 rows with a similar configuration on a DB2 Version 8 database. This is a performance improvement of almost 19 times!

The tableBASE table would be cumulative so that the final tableBASE table is equivalent to the 'heap' table.

The benefit of using tableBASE tables for 'heap' tables is realized not only when the data is written 19 times faster than DB2, but also when the 'heap' tables are processed, as the rows can be read 22 times faster than DB2.

DB2 Stored Procedure Support

DB2 on the mainframe is often used as a high performance backend database server, especially for web applications. These web applications often make use of DB2 Stored Procedures to improve performance. Even greater performance improvements are possible through the use of tableBASE by keeping read only tables memory resident. Virtual Table Share tableBASE Regions (VTS-TSRs) must be used to share these memory resident tables between all Stored Procedures.

Stored Procedure access to tableBASE tables is provided by the native tableBASE application program interface (API) or through DB2 User Defined Functions (UDFs). The DB2 UDFs for tableBASE tables are considered table functions and can be used in any SQL statement, like any other table function. Details on how to write UDFs for tableBASE tables can be obtained from DataKinetics.

Boosting Transaction Processing

Mainframe (CICS, IMS, WebSphere) transactions are often used as the backend business logic layer between a WebSphere application server and the DB2 database. Keeping all or some of the business logic on the mainframe may be related to available CPU capacity, skill levels of staff, or simply providing a web interface to existing transactions.

tableBASE can be used to boost the performance of mainframe transactions by providing up to 2 Gigabyte address spaces for read only data tables at mainframe memory speed.

Conclusion

The technical architecture of any high throughput or limited batch window DB2 application should be reviewed to determine how tableBASE can be used to boost throughput or reduce the batch window. Although tableBASE is well suited for new development, the performance of existing applications can also benefit, particularly if a major hardware upgrade is the only other alternative.

Application controlled parallel processing, using multiple instances of the same batch program to process partitioned data, is one way in which mainframe hardware can be utilized to maximize throughput or reduce the size of the batch windows. Without tableBASE, each instance of the batch program would have its own copy of read only tables in its working storage, resulting in large programs and wasted space in memory. With tableBASE, each instance of the batch program would share a single copy of the read only data.

In addition, if multiple passes of the same data is required, as would be the case when the data is partitioned, and then processed after it has been partitioned, tableBASE can be used to build memory resident tables that can then be used in the second pass of the data, to further reduce DB2 data access overhead.

If an organization is using DB2 on z/OS for batch processing, one might want to consider how tableBASE can be adapted to the batch architecture to boost performance. The latest DataKinetics/IBM benchmark results, available from DataKinetics, can be used as a guide in determining the potential savings in CPU overhead. A cost/benefit analysis would then determine whether tableBASE is a solution for the organization's performance issues.

Detailed information on the benchmark tests is available upon request. Contact: Keith Allingham at (613) 523-5500 x230, kallingham@dkl.com.