Tuning PeopleSoft Applications for the DB2 For OS/390 Environment

PeopleSoft provides a sophisticated suite of Enterprise Resource Planning (ERP) business applications. On OS/390, the most commonly used applications are human resources and financial systems. These business applications, especially when using Version 7 and later releases, use a complex 3-tier client/server architecture.

While many performance problems involve network and client-server tuning issues, a large opportunity still exists for the OS/390 database administrator (DBA) to improve DB2 performance. The focus of this paper is host DB2 based tuning, and is based upon experience at several client sites.

The Nature Of PeopleSoft Applications

Many applications have requirements that increase the amount of DBA support needed. For the last several years, these are:

1. Large tables
2. High availability
3. High transaction volumes

These traditional dimensions of complexity are often found in combination. For example, a VLDB may have a large user community and be needed on a 24 hour, 7 day per week basis.

ERP applications have added another dimension of complexity to everyday DBA life: a tremendously large data model. Depending on the release and the number of ERP applications being implemented, this is at least in the 2,000 table range. Full implementations of PeopleSoft 7.5 have more than 5,000 tables, while SAP's latest release has more than 13,000 tables.

ERP applications are strategic and, even though they are purchased applications, require a significant effort to be successful. ERP applications consist of several application components, and some or all of these may be completely or partially installed. Even after the initial production implementation, the ERP environment will grow and the performance characteristics will usually change significantly. Your tuning opportunities will change and perhaps grow over time.

An ERP architecture provides the business user with a GUI based front end that is much more attractive and easier to use than many of the older transaction based environments that are being replaced. The applications are easily tailored to the requirements of an organization because a table-driven architecture is used. During the implementation process, the business designers at an organization modify and design screen and report layouts, as well as the overall processing and system flow. In PeopleSoft, these designs are stored as data within DB2 in the PeopleTools tables.

These applications run on many hardware platforms, operating systems and DBMSs. As stand alone applications, many companies implement these products on a non-mainframe platform. As many large IBM customers adopt these packages, they are implementing them on their corporate standard database: DB2 for OS/390.

There are many challenges facing the OS/390 DBA. Since PeopleSoft operates on many different platforms and can use a variety of database management systems (DBMS), the physical design has not been optimized for all DBMS environments. As an example, using DB2 for OS/390, PeopleSoft is delivered as one database and a small number of table spaces that contain 1,700 to 2,500 tables. The number of tables grows with each release and Release 7.5 contains over 5,000 tables.

There are a myriad of issues that must be addressed to correctly break this into manageable pieces; these range from allocating objects across physical storage to determining which objects to split off into other databases (to reduce DBD size) and table spaces. The tuning of the application's physical structure is essential, *both for performance and manageability*.

PeopleSoft/DB2 Performance Issues

In many PeopleSoft/DB2 environments, several performance problems exist in the layers of software and network between the PC and the host DB2 system, as illustrated in Figure 1. These problems require a good level of coordination and teamwork between PC, LAN, network, OS/390 systems programming, and DBA staff.

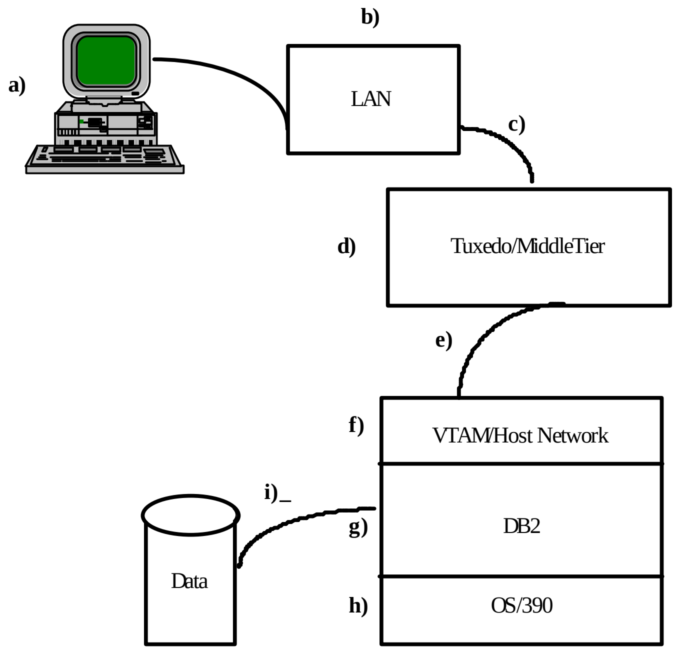

FIGURE 1. Areas that require Performance Tuning

Performance issues may exist in the following areas:

a) PC problems: CPU speed, memory management or PC based caching
b) LAN to the gateway
c) Gateway performance or router
d) Tuxedo tuning on middle tier
e) WAN connection from middle-tier to host DB2 system
f) Network performance tuning (delay parameters, buffer sizes, VTAM parameters)
g) DB2 performance on host
h) OS/390 tuning
i) DASD subsystem tuning

All of the data is stored in DB2 on the host, and all SQL executed by DB2 is dynamic. Some SQL for the PeopleSoft provided COBOL batch programs is stored in PeopleTools tables, while other SQL, particularly PS/nVision, is built on the workstation and sent to the host for execution as dynamic SQL. This requires special care and does not easily lend itself to many of the traditional methods of application tuning.

There are several DB2 tuning techniques that must be used to improve performance. These include changing buffer pool sizes and object allocations, altering the index design structures, altering DB2 subsystem parameters, data compression, enabling the DB2 Resource Limit Facility, and possibly changing the OS/390 dispatching priorities for the user and system address spaces.

The nature of the DB2 environment and PeopleSoft allows for most changes to be done quickly. Buffer pool size and threshold changes can be made at any time while DB2 is operational, and may need different sizes and thresholds for online and batch processing environments. DB2 uses additional buffers immediately, or flushes buffers if the allocation is reduced. Most index changes can also be made daily as there are no plans or packages to be rebound as part of this process. Some of the typical benefits of performance changes are quantified and discussed later in this document.

Summary of DB2 and OS/390 Performance Issues

This section briefly summarizes areas where performance adjustments were or can be made, and a more detailed description of each performance task, method, and its rational are provided later.

1. Buffer pool tuning

   The best method for reducing I/O, its associated CPU time, and reducing the elapsed time for transactions and large queries is through buffer pool tuning. The proper sizing of the buffer pools, setting of thresholds, and the *proper grouping of objects into multiple buffer pools* can provide significant overall improvements. Key tables and indexes must be properly grouped into separate buffer pools.

2. Index adjustments

   The application is delivered with indexes that support updates, but do not necessarily support all the various queries necessary for the business requirements. New indexes were added to support query processing, and were further adjusted for efficiency. For example, some of the tables to consider for PeopleSoft financials index changes are the Ledger, Journal Account Line, Vendor, and Voucher.

3. OS/390 dispatching priorities

   Dispatching priorities for the primary DB2 address spaces should have the relationships as recommended in the DB2 installation guide. These address spaces should not all be at the same priority and must be higher than the user/application address spaces.

4. Governor for ad hoc queries

   DB2 includes the Resource Limit Facility to cancel dynamic queries that use more than a specified amount of CPU time. This can be used to control runaway PS/nVision queries. One drawback to this governor is that it does not have any knowledge regarding how close a query might be to finishing, and it may cancel a query just moments before completion. The setting of threshold limits is somewhat subjective and may become a political issue within the organization.

5. Network performance review

   There are always concerns about elapsed times, both for transactions and queries. In tuning the PeopleSoft environment, large gains may be found in Network tuning. Network performance may vary substantially between locations depending on the network configuration. The tuning effort must be a team effort by all departments who share the responsibility for various components. Monitoring devices and components should be used at various points on the network.

6. Hardware data compression

   Many PeopleSoft tables contain patterned data that can achieve a high rate of compression. Compressed data means that more data can be stored per page, providing better buffer pool performance and fewer physical I/Os. Certain tables, such as the Ledger table, obtain a good benefit from compression.

Performance Measurements

Figure 2 summarizes a tuning effort for a production PeopleSoft environment with PS/nVision queries. These queries consume most of the resources, and show large improvements from tuning. At the end of the tuning effort, significantly more users added to the system with no impact on the original existing production users.

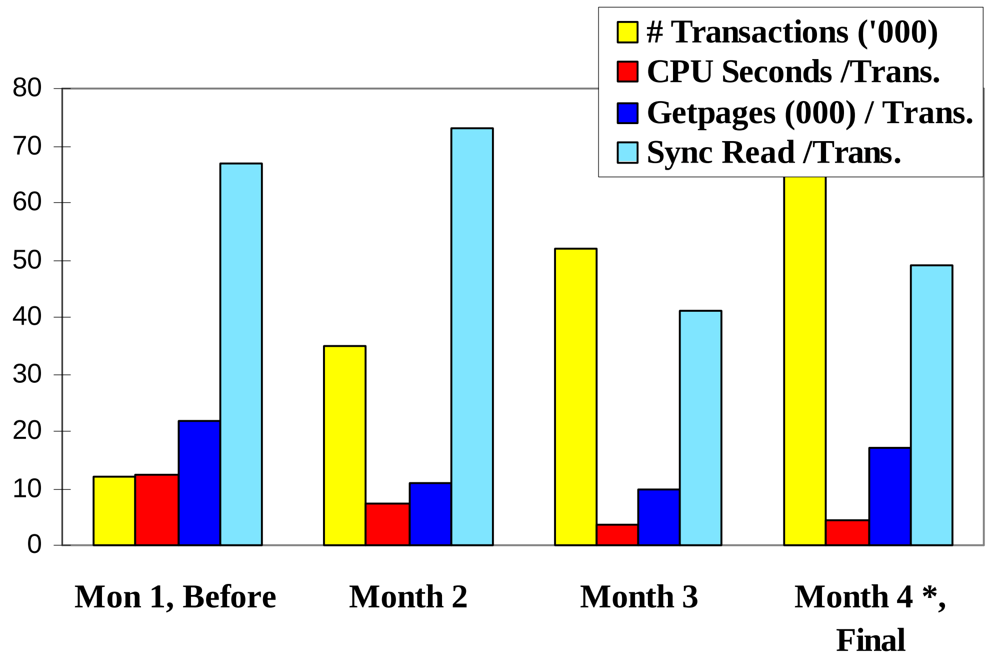

FIGURE 2. nVision Reporting

* Month 4 is based only on the first week of the month. Projecting this out to a full month shows a continued rate of growth for the application. The average CPU utilization for Week 1 of Month 4 is higher than Month 3. This is caused by the heavier application demands following month end. Month end reporting involves a large number of PS/nVision queries that perform a great deal of summarization activity against the database. Since other processing does not require as much CPU processing, the average CPU cost per transaction over the month should not be any heavier than the third month. Extrapolating the transaction volume for Month 4 shows another increase of volume compared to Month 3.

The first tuning in Month 2 involved some high level tuning of buffer pools, and object placement within buffer pools. The PeopleTools were some of the first tables placed in their own buffer pool. Other changes were based on my experience of tuning non-PeopleSoft environments.

In Month 3, improvements included changing indexes and the use of Buffer Pool Tool to perform more refined object placement and modeling of buffer pool size. These changes further reduced the CPU, getpages, and the synchronous I/O. No further changes were made before obtaining the measurements for Week 1 of Month 4.

It must be noted that there are many factors that affect the purity of these numbers. History was not completely separated by day for month 2, so the first tuning effort may have resulted in better results. The effect of a CPU upgrade was removed from the figures. Several changes were made to buffer pool sizes and object allocations, and the effect of the individual changes were not captured.

New departments were added each month. Some new users had spikes in their individual use, because of a learning curve. One user, when contacted about an exceptionally heavy usage the day before, said

Oh, I tried to cancel that.... The first week of each month shows much higher usage due to PS/nVision reporting.

Further changes should be made by continuing the restructuring of indexes. In an application that already contains thousands of indexes, the necessity to add more seems somewhat ironic!

The remainder of the application consists of batch jobs and utilities. Most of the updates made to the application are performed by batch jobs. The utilities are used for taking backup copies and reorganizations.

Many performance problems may worsen as the user load increases. If a phased implementation is used, problems may appear as the application is rolled out to the various departments. To avoid this possibility, stress testing should be undertaken with a representative load prior to the production date.

One of the major performance concerns at another organization was voucher processing. One or two nights per month, the processing would be very heavy and the job would run for hours, often being canceled after exceeding 7 hours. As volumes increased in late July and August, something needed to be done. Buffer pool tuning and reclustering one table brought the elapsed times down, even with increasing volumes. The results are shown in Figure 3.

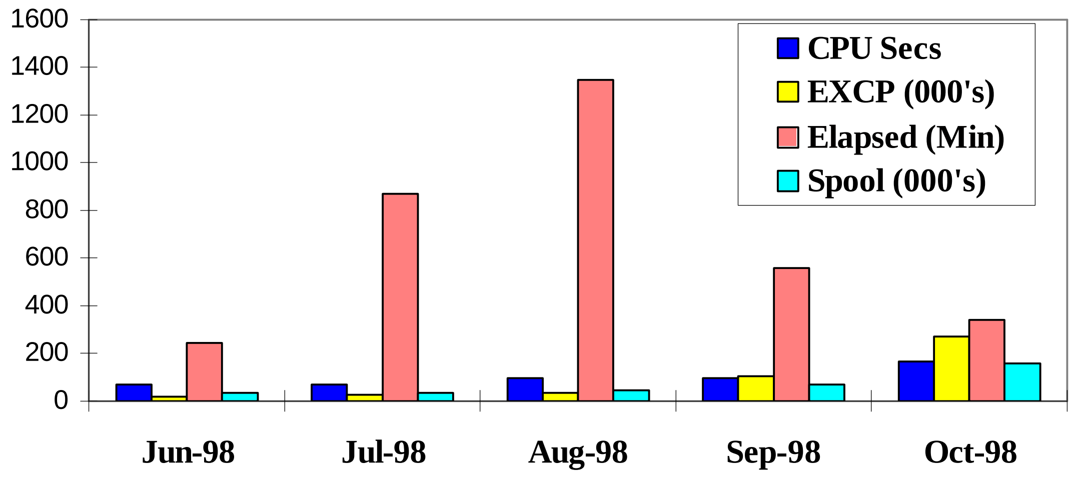

Figure 3. Voucher Processing

After PeopleSoft Release 6 was put in, the clustering change was lost, and performance remained acceptable. Shortly thereafter, an SQL patch was received from PeopleSoft to fix an a specific SQL error that occurred on certain addresses. This patch reintroduced the original performance problem. After the re-applying the clustering fix for the table, performance was restored.

Description of Performance Activities

Buffer Pool Measurements

Buffer pool tuning is one of the major methods for tuning the PeopleSoft application on the host. The tuning exercise is generally done in the production environment where the workload to be tuned resides. The immediate benefits of buffer pool tuning are improved on-line response times and batch run times realized by users of the DB2 system, improved user productivity, greater system throughput, reduced CPU utilization, reduced I/O workloads, and the optimization of DB2 memory resources.

DB2 performs input/output (I/O) on behalf of its users. This can involve a great deal of disk activity to retrieve data. This activity is called physical I/O. Physical I/O causes delays in processing and uses other computer resources such as CPU to process the requests and the data returned. Physical I/O is one of the more expensive operations that computers perform in terms of resources used and delays in getting results.

To avoid physical I/O, DB2 uses computer memory, called buffer pools, to store frequently accessed data. Data can be made available more quickly when the data is found in the buffer pool for a subsequent application request. A request for data by an application is called a logical I/O or getpage. A getpage may or may not result in a physical I/O depending on whether the page is found in the buffer. DB2 can use up to 60 separate buffer pools to which DB2 table spaces and indexes can be allocated.

Figure 4 shows where logical I/O (GETPAGE) and physical I/O is performed.

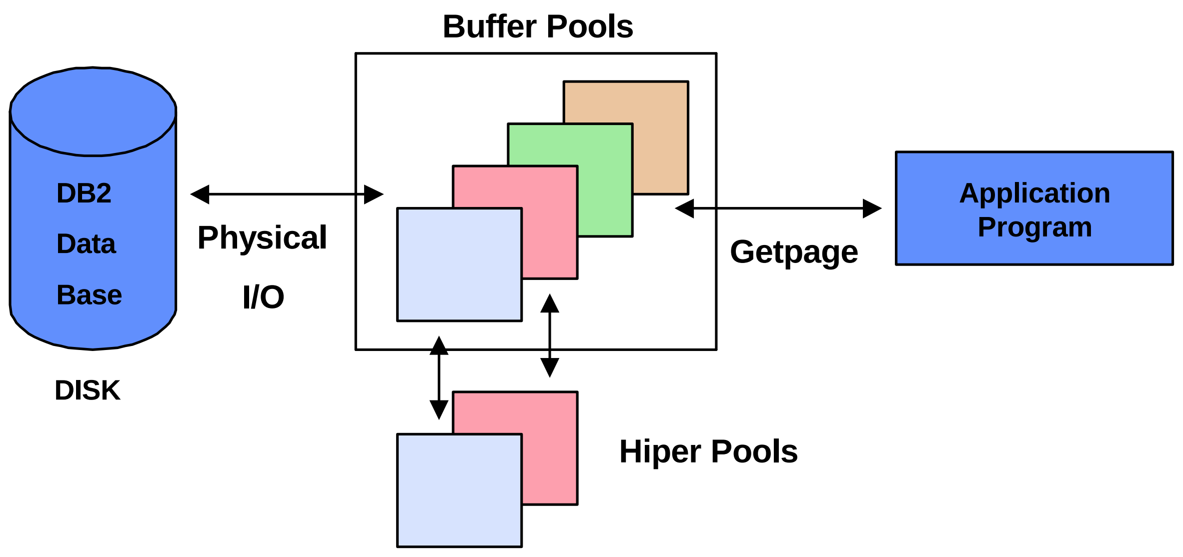

FIGURE 4. Buffer Pools

There are two major types of I/O performed by DB2. The first type is random I/O. This means that a given piece of data is accessed by DB2 using an index to find the appropriate page and retrieve the data required. Random access is efficient and is even better when no physical I/O is required to satisfy the request.

The second type of I/O is sequential (SP). As its name implies, sequential I/O requires that all or part of a table space or index is read sequentially to find the data required by an application. This usually means a lot of I/O and processing by DB2 to find the result requested. In certain cases, such as the processing of an entire table by a batch job, sequential I/O can be of great benefit as the pages will be found in the buffer as needed, reducing or avoiding application I/O wait time.

For 4K buffer pools that are 1,000 pages or larger, DB2 will read 32 pages in one prefetch I/O for SQL and 64 pages for utilities. For 32K buffer pools, 8 times more buffers are used than for 4K pools. Sequential prefetch works so well that two additional types of prefetch were added in addition to the original sequential prefetch; List (LP) & Dynamic. SP and LP may be determined at BIND time or at execution time for dynamic SQL. Prefetch causes some processes that might be bound by I/O to become CPU bound. This means that the CPU speed becomes the limiting factor in system throughput rather than the wait for I/O to occur.

List prefetch is like skip sequential processing: a list of Row IDs is built from one or more indexes and sorted, before the data is retrieved in the order it is stored on DASD. Up to 32 data pages can be read (based on prefetch quantity) before the first row is returned to the application program. DB2 triggers sequential prefetch for the initial set of pages read. Both pure sequential and list prefetch are shown on the Explain output.

Dynamic prefetch, or sequential detection, is a run time decision by DB2 if the processing finds a sequential pattern in the data access. Monitoring continues, and if the pattern changes to a point where sequential prefetch is not helping, it is turned off. This can happen multiple times in one execution.

Sequential I/O uses a lot of buffer space and can lower the probability of finding a page in the buffer for random request. Note that any type of prefetch may indicate a poor access path for on-line or ad hoc processing. Prefetch will also reduce the space available in the buffer pool for randomly accessed pages.

Starting with DB2 V3, DB2 can use hiper pools to back the virtual buffer pools. This allows additional memory in expanded storage to be allocated for data storage to avoid physical I/Os. Expanded storage will provide performance benefits if a page is found there and a physical read is avoided.

Hiper pools (HP) are slightly slower and more expensive than reading directly from the virtual pools. When HPs were introduced, expanded storage was cheaper than central storage. With a 9672 processor, all memory is the same and the system programmer decides how much will be central or expanded. When asked, I encourage the central storage to be made larger and the expanded storage smaller. Nobody wants to set expanded storage to zero, since both some sort functions and OS/390 requires some expanded for processing. On a 9672 processor, HPs should only be used when more than 2 Gigabytes of memory exists.

It is critical to have sufficient real storage available to avoid OS/390 paging. Paging allows users to share resources by swapping pages of memory to expanded storage or DASD. DB2 physical I/O will be reduced, but all users, including non-DB2 users, will get poor performance if the system paging rate is high.

The fact that DB2 has fifty 4K buffer pools lends one to think that splitting DB2 objects into separate buffer pools would be a good thing to do. It is important to identify I/O activity, type and rate of access, and reference patterns for DB2 objects. Objects having poor access methods (such as frequent large scans) should be identified and steps taken in the application to improve performance or be moved to another pool to reduce the impact to on-line transactions.

Key measurements, called the *system hit ratio* and the *application hit ratio*, are determined by the number of requests for the page (getpage) and the number of times a DB2 page is found in the buffer pool.

The application hit ratio is typically used by on-line monitors, and is computed as:

getpages / read I/Os or   (getpages - read I/O) / getpages

This formula is useful only for application performance where almost all access is random. Most application workloads have both random and sequential access to data. When this is true, the following *system hit ratio* provides a more accurate measurement: of both system and application performance.

(getpages-pages read) / getpages

The system hit ratio is a more meaningful measurement because it shows the request for pages and the number of pages that must be read from DASD to satisfy the request. Asynchronous I/O via sequential prefetch can have a negative impact on buffer pool efficiency. It is possible to have a relatively high application hit ratio while also having a low system hit ratio.

Sequential I/O lowers the possibility of finding a page in the buffer for another request. Even if a large buffer size is used, there is little chance of a page still being resident for another application request. A pool of 1,000 to 5,000 buffers is often adequate. Separating sequentially accessed objects into a buffer pool away from randomly accessed objects will improve the performance of the random objects.

To obtain the best tuning results, it is important to tune for the peak periods for application use. Proper buffer pool sizes, combined with the proper grouping/allocation of objects to multiple pools, can make dramatic improvements to application performance.

Buffer pool tuning is a major method for tuning the PeopleSoft application on the host. The ***Buffer Pool Tool from Responsive Systems*** provides detailed tuning and modeling support that tracks I/O rates by dataset for both table spaces and indexes. Using its modeling facility, the buffer pool sizes can be determined quantitatively and verified, as well as the effect of re-grouping the objects into different pools. The software provides both a tuning methodology, and a clustering technology, to help determine which objects belong together in a pool.

Buffer pool tuning is not a "one time" exercise. The buffer pools must be monitored on a regular basis and re-tuned as the workload grows, as new users begin to access the system, additional PeopleSoft function is implemented, or the system configuration changes.

Buffer Pools and PeopleSoft

Many organizations start out using four or fewer buffer pools. As an example of a four pool starting configuration, BP0 is used for the catalog and directory, the temporary work database is in BP1, and the table spaces and indexes are in BP2 and BP3 respectively. For illustrative purposes, this is a reasonable starting point and provides good tuning opportunities in a PeopleSoft environment.

In tuning PeopleSoft, the two main objectives are to separate sequentially accessed objects from randomly accessed objects, and to isolate key tables and indexes into separate buffer pools. After the tuning exercise, there will usually be six to eight buffer pools in use.

The first task is the identification of the most frequently accessed read-only tables, and placing these in a separate buffer pool. The working set size of each dataset should be examined with a goal of sizing the buffer pools so the most frequently accessed pages of these tables remain resident in memory (page fixing in OS/390 terms). As a starting point in the tuning study, using two additional buffer pools for the table spaces and indexes respectively allows these objects to be resident. After making these changes, an analysis with Buffer Pool Tool and the on-line monitor showed this to be true.

Key tables to isolate for nVision are the PeopleTools tables. These tables can be identified by the missing underscore that is normally in the third character of their name. The PSTREE tables are very good candidates. Other good candidates are tables with high read and low write activity, such as the Ledger table. The following table shows buffer pool changes during a tuning exercise. In this example, there is still more work to be done as more users are added and application function is implemented.

In Figure 5, the results of a buffer pool sizing exercise is shown. The application demand for data also increased as shown by the increase in getpages. Although it is not shown, the overall performance of the application improved even with the increased workload. Note that BP4 was defined, but no objects have been moved to that buffer pool.

FIGURE 5. Buffer Pool Sizing

BP Name	VP Size August	Getpages August	VP Size October	Getpages October
BP0	1,000	2.47M	3,000	0.39M
BP1	5,000	0.83M	15,000	3.67M
BP2	7,000	0.73M	7,000	0.07M
BP3	7,000	0.06M	7,000 HP 30,000	5.28M
BP4	0	0	7,000	0
BP5	0	0	7,000	0.78M
Total	20,000	4.09M	46,000	10.2M

Of course, life gets interesting if an object is accessed both sequentially and randomly. Perhaps index tuning to modify the sequential access paths will help. Also, it may be possible to modify buffer pool allocations at certain times of the day to improve DB2 performance when other workloads will not be affected.

Buffer Pool Tool can be used to gather more detailed statistics and to model quantitatively how large the buffer pools should be. This product showed substantial benefit would be gained by increasing BP2 and BP3. This change improved the hit ratio in these pools and significantly lower the I/O rate. The hit ratio is the number of requests for the page (getpage) divided by the number of times a DB2 page is found in the buffer pool. A getpage is a logical read.

Another key feature of the Buffer Pool Tool is the ability to show both detailed access and I/O at the dataset level. This feature shows whether the I/O is sequential or random. Random I/O normally means a request for a small amount of data where an index is used. Sequential I/O normally means that a lot of data is accessed and many I/Os are performed, presumably due to the inefficient use of an index or by performing a scan of many or all of the pages in a table space or index. To solve these problems, changes were made to the indexing strategy.

It should also be noted that the data collections for Buffer Pool Tool were done using one hour snapshots of activity during busy periods of the day. We did notice that some tables are used more heavily at certain times of the month. It is possible, although somewhat unlikely, that other times may have shown quite different results. The results that we achieved met the expectations of I/T staff familiar with the application.

Until growth of the PeopleSoft user base at customer sites is complete and the entire application is implemented, buffer pool tuning will be a key factor for achieving optimum performance of this application. It is recommended that the data collection be done weekly during this period. When the environment has stabilized, monthly collection will provide trend analysis and help with the early identification of performance problems. Buffer pool data collection should be done regularly to check usage patterns and possibly modify buffer pool sizes and assignments.

Physical Design Review

The main subject of DB2 physical design is index placement and key columns. This is particularly important with PeopleSoft since the SQL and table design cannot easily be modified.

Indexes in DB2 are used to enforce uniqueness of column values and to reduce processing costs. Processing costs may be the access time required to retrieve data or the CPU time to perform a sort. While properly designed indexes help in these areas, poorly designed indexes may not provide help performance, and can even add significant processing time and costs. Fortunately, the DB2 optimizer usually avoids indexes that do not save processing time. It is important to understand that every index adds some processing overhead for update functions, so unused or infrequently used indexes should usually be eliminated.

In correcting index problems, the first temptation is to add more indexes, and place additional columns in the index. These solutions easily lead to overkill. A lot of the tuning effort for PeopleSoft is the restructuring of indexes. Mainly this has involved removing indexes, reducing the number of columns in an index, or reordering columns. On occasion, indexes were added or columns were added to an index. The sort sequence of columns should also be considered. For example, information retrieved most often in a reverse chronological order can be placed in descending order.

Another problem with the delivered PeopleSoft design is the clustering index is normally placed on the primary index. The primary index is used to ensure unique values of the primary key so that data integrity is assured. Where the primary key is used for inquiry, a single row is normally expected. This means it is not usually a good choice for the clustering index unless the table in question is often processed in primary key sequence. It is often better to make the clustering index one where the cardinality (FULLKEYCARD) is low or range predicates are used in the WHERE clause.

The clustering index is an important facility for returning an answer set for a query where more than one row is expected. If more than one row can be found on the same page, or in close proximity, there can be a substantial saving in I/O and CPU processing. A common example relating to PeopleSoft is retrieving ledger entries by accounting period and fiscal year. The primary key on the ledger table has these columns too far down in the index for it to be used for queries of this type. It must be remembered that there may be some tradeoffs between the relative importance of certain queries over others. On some tables, the choice is easy; on other tables, the choice is not quite as clear.

The majority of index changes can be done without a significant outage. In the PeopleSoft environment where dynamic SQL is used exclusively, the DDL to drop and recreate a index definitions can be done any time during periods of low activity. RUNSTATS may also be run, but this runs quickly for the indexes alone.

Changes to index clustering require more time and planning. Following the dropping and recreating of at least two indexes, it will be necessary to reorganize the table space to place it in the order of the new clustering index. This will take more time and care than just changing the columns within the index.

The largest savings in CPU cost will occur by buffer pool tuning accompanied by improving the physical design to support the SQL generated by PeopleSoft. While modifying the SQL would be ideal, there is still a lot of benefit that can be obtained by reviewing the indexes on their own merit.

Tuning SQL

PeopleSoft stores all data on the DB2 server. All SQL is either stored as statements in within a DB2 table, or they are built on the client workstation in response to ad hoc requests. This type of SQL is called dynamic SQL, as it is bound (compiled) by DB2 immediately before execution. This approach has several advantages and disadvantages associated with it as discussed below.

With the SQL being generated on the workstation for each execution, tuning SQL for this application is not practical. This is a clear disadvantage if changes to the SQL are desired. It also eliminates from consideration denormalization of tables or redistribution of columns, since changing the SQL would be extremely difficult.

The use of dynamic SQL is both an advantage and a disadvantage. There is a cost associated with SQL preparation through the bind process. This cost is incurred for every SQL statement within the PeopleSoft application. Where repetitive execution of the same statement occurs, the bind time can represent a significant cost. Since the SQL is not saved between executions of a statement, it is impractical to spend time changing SQL, unless a significant problem is found in the SQL generation process. This type of problem would have to be reported to PeopleSoft for resolution. If static SQL was used, the access path would be saved and be available for the next execution.

Based on the nature of many PS/nVision queries, dynamic SQL may provide performance benefits over static SQL. Static SQL must make assumptions about the values entered by the user, whereas dynamic SQL has all of the values specified in the WHERE clause. An example of a dynamic SQL predicate is:

```
VENDOR_NAME_SHORT BETWEEN  ACME AND ACME
```

This example should return one or very few rows to the user. By contrast, static SQL would not know the values for the BETWEEN when the access path is chosen, and would make assumptions that may not perform as well as having the known values in the dynamic statement. If the cost of the bind is small in comparison to the execution cost, dynamic SQL is preferable.

This statement also shows why PeopleSoft does not benefit as much from the DB2 V5 dynamic statement caching as some other ERP software. PeopleSoft currently uses literals in the generated SQL. As the literals change for each new SQL statement, DB2 cannot obtain good benefit from cacheing the SQL statement for the next execution. This will be fixed in Release 7.5 of PeopleSoft when parameter markers will be used instead of literals in the generated SQL. There can be large benefits with DB2 V5 caching for batch SQR reports in PeopleSoft releases before 7.5.

Capture and Explain SQL

A PeopleSoft trace on the client can be used to capture the generated PS/nVision SQL. These statements can be manually explained. A result of this effort might be adding some additional indexes to some tables.

Another technique can be used later to also capture SQL and its access path. An on-line monitor can capture the dynamic SQL and the mini-plan generated by DB2 (IFC IDS 22 and 63). The mini-plan is similar in content to explain output. This involves a real time trace, that should only be turned on for a very few users to minimize the impact on system performance and the number of records generated.

The mini-plan is valuable for determining which indexes are used on some of the key tables. This method is useful to confirm if changes to indexes are working as designed.

Data Compression

In the earlier releases of DB2, data compression was considered to be a trade-off between saving DASD and the use of CPU to compress and decompress the data. Unless the compression rates were sufficiently high, data compression would not provide sufficient benefit to make it a viable choice. In many cases the cost of compression is offset by the CPU savings of doing fewer I/Os as a result of having more rows of data per page, and thus fewer pages. Another factor that has to be considered in this area is the type of processing for the data - if a lot of the data is actually accessed, as in a tablespace scan, then the decompression cost can be high.

The newer processors and DB2 V3 introduced a hardware data compression feature for table spaces. Indexes are not compressed. Using microcode to perform the compression and decompression, the impact on CPU time is minimal. The compression algorithm is quite good for most types of data. It uses a compression dictionary to store the most frequently found patterns within a table space, which is similar to the popular PC program PKZIP.

Many PeopleSoft tables contain patterned data that should achieve a high rate of compression. Some tables, such as the Ledger table, obtain good benefits from compression. Other tables, such as Journal Line, did not provide any benefit from compression in the testing.

The utility DSN1COMP can be used to check the effectiveness of compression to choose candidate table spaces. On the Ledger table, it showed a 50% compression rate would be achieved if this table was compressed. Data compression should be used in the test environment to gain familiarity prior to using it in production.

This feature is turned on using the COMPRESS YES parameter on the table space followed by a REORG. If a table space does not get a good benefit, it can easily be changed back to COMPRESS NO. There is a minimal risk for compressing a table space. If you achieve good benefits, fine. If you don’t, the additional resources will be less than 5 per cent for that object until you can return it to an uncompressed state.

Network Performance Review

Network tuning for PeopleSoft could easily be another white paper, with many more topics than are included for this discussion of DB2 tuning. This will require coordination and teamwork between PC, LAN, network, OS/390 systems programming and DBA.

Monitors can be used to determine if significant amounts of time are being spent on the network. It should be noted that the PeopleSoft design requires a large number of short conversations across the network, and each conversation has some network overhead associated with it. High elapsed time in DB2 monitor reports may also be due to thread reuse.

Other Performance Areas

1. Investigation of Hiperpools
   Hiperpools support virtual buffer pools and reside in expanded storage rather than main storage. Expanded storage is generally used for OS/390 paging and the few other applications that can use it, such as DB2. Hiperpools should be expected to be more volatile as OS/390 will normally have priority over DB2 for paging purposes. In DB2, a hiperpool augments an existing buffer pool to support larger memory for buffers. Hiperpools have helped some organizations where there is a large amount of read-only activity.

   The use of expanded storage for hiperpools was especially attractive when expanded storage was substantially less expensive than main storage. However, this is no longer the case. One type of memory is used for main and expanded storage on CMOS processors; the organization determines how much will be used for each purpose. Logical I/O is still faster than hiperpool I/O. When configuring a CPU or memory upgrade, allocate more storage as main and less as expanded. This additional storage, if available for DB2, should be allocated as virtual rather than as a hiperpool.

   Using large virtual buffer pools is generally preferred, provided there is sufficient memory available to avoid excessive OS/390 paging. If the overall workload increases to a point where OS/390 paging becomes a problem, it may be necessary to reduce the size of the virtual buffer pools for the PeopleSoft application. At this point, it may be possible to maintain DB2 performance by using hiperpools . The CASTOUT attribute for the hiperpool should be set to YES to allow OS/390 to have priority over DB2.

2. Use of the Resource Limit Facility
   The Resource Limit Facility provides a control for runaway dynamic SQL. It sets a maximum amount of CPU time by application or by user. The limits can be changed dynamically at any time while DB2 is up. The facility is now being used in the test region.

   There are also PeopleSoft controls for the number of rows returned and the number of tables joined. These controls have not been implemented at this time.

3. OS/390 Tuning
   Discussions with OS/390 personnel regarding dispatching priorities for the DB2 address spaces may be necessary. All the DB2 address spaces, the IRLM, MSTR and DBM1,should be set higher than other production workloads. These address spaces perform vital system functions and other tasks such as locking on a frequent basis but use little system resources. Most of the processing cost is charged to the user address space, and almost all functions execute at the dispatching priority of the user address space.

Summary

PeopleSoft is representative of the new wave of advanced applications that business users are demanding. Having bought the software, DBAs and other technical people are expected to make it operate effectively in their environment.

The client-server tuning issues sometimes obscure the many opportunities available for DB2 for OS/390 tuning. The experienced DBA should consider the corporate and client benefits of DB2 tuning for performance, and use tools and methods to reduce the amount of physical I/O performed by PeopleSoft. Tuning of the application on the mainframe, and the DB2 for OS/390 system will provide substantial paybacks in the areas of:

- Elapsed time reductions for on-line and batch processes
- Reduced processor CPU consumption
- Reduced operating cost of the application

Martin Hubel Consulting Inc.

Martin Hubel Consulting Inc. is a provider of computer consulting services and education specializing in the strategic technologies of relational database management systems, project management, and advanced systems development methodologies. These technologies help customers meet business goals that include competitiveness, productivity and cost-effectiveness.

Martin Hubel has extensive experience in database design, application architecture, and system administration for relational database management systems, particularly DB2. Over the past seven years, he has advised and taught more than 150 clients worldwide to use DB2 effectively, and designed DB2 product lines for software vendors. Prior to his software experience, Martin worked in both the private and public sectors as a project manager, database administrator, systems programmer, security and disaster recovery analyst, application designer, business analyst, and end user consultant. He has been active in a number of professional organizations including GUIDE International and the International DB2 Users Group and is an internationally recognized speaker on various topics relating to DB2. Martin's current clients include companies in the finance, communications, health care, public utilities, and software industries.

Martin’s consulting and teaching activities also include DB2 on the NT and AIX platforms. As with his other consulting work, it focuses on design, performance and systems management issues. He is a member of the DB2 Gold Consultants Program run by IBM. He contributed two chapters to a book published by Prentice Hall entitled: "Data Warehousing: Practical Advice From the Experts."

About Canadian Tire Corporation

With 430 stores, Canadian Tire is the premier Canadian retailer of hard goods. Having celebrated their 75th anniversary last year, they can be considered a national institution. Their products and services include automotive, home, sports and leisure, gas bars, and service centers. Gross operating revenue exceeds $4 billion, and their Associate Stores employ 34,000 people. When Canadian Tire wanted to replace their financial systems, the size of their operation required a large enterprise style solution. After a selection process, they chose PeopleSoft and have been in production since November 1997.

Distributed by:

Responsive Systems
281 Highway 79
Morganville, NJ 07751
(732) 972-1261
(800) 322-3973
(732) 972-9416 Fax

Email: db2xpert@superlink.net

Web Site: www.responsivesystems.com