Transaction processing in the reservation industry

by JAMES R. MANCHESTER
Quality International Inc.
Scottsdale, Arizona

ABSTRACT

This paper presents some background on transaction processing relative to the airline and hotel reservation industries. It discusses how hardware and software have changed to be compatible with changes in application priorities. Finally, some of the current stumbling blocks to successful transaction processing systems are identified.
INTRODUCTION

Transaction processing is a term used to describe an on-line, real-time computer system whose prime characteristics are excellent terminal response time, availability, reliability, and recoverability. Each of these terms has specific meaning in this segment of the industry.

1. Response Time: Ninety percent of the responses should occur within 3 seconds or less.
2. Availability: The system should be available in excess of 99% of the scheduled operating hours, which may be 24 hours per day, 7 days per week.
3. Reliability: The system should process messages without errors and with consistent results.
4. Recoverability: The system should recover from most errors without affecting the user or corrupting data or provide alternate means of recovery when this is not possible.

HOW IT ALL BEGAN

Transaction processing in the airline industry was initiated in 1958 by the joint development of a centralized, on-line, real-time reservation system for American Airlines using an IBM 7090 computer and IBM 1301 disk files. On-line implies operational and available for use. A good definition of real-time is assured, nondeferrable, on-demand servicing of unscheduled, unpredictable, diverse requests in an environment of fixed resources. The intent was to provide both a centralized reservation industry and access to the reservations by agents at the airports, thus giving the passenger the ability to create or modify a reservation at any time from any place. A significant feature in this system was that the name and other pertinent information about the passenger was maintained with the reservation. This also required considerably more storage space for recording the information.

To maintain the Passenger Name Record (PNR) and meet the response time criteria, a specialized reservations program had to be developed, since a general-purpose operating system could not handle the number of messages within the desired response time requirements. This program consisted of a complete operating system as well as an application system tailored to the airline reservations requirements. Pan American, Delta, and American Airlines each tackled the problem in their own way, but jointly with IBM. Each developed a system on a different IBM mainframe. The three systems were all operational by 1964, and performed according to design criteria.

In the sixties, the airlines were growing rapidly. It became obvious that the systems could only handle a specified number of transactions per second before a faster system would be needed. It was also apparent that other airlines would require similar reservations systems. Hence, in 1964, IBM embarked on a new airline reservations system which could be used by any airline. They named the product PARS (Programmed Airline Reservations System). This generic reservation system was designed to operate on the System/360 hardware. It provided a solution that could be purchased by much smaller airlines while providing a growth path for them. Even the largest airline at that time would be able to operate on this family of equipment with the new software.

PARS was the name of the complete package of software that IBM provided the customer. This package consisted of programs performing seat inventory, seat availability, name lists, schedule changes, passenger data entry functions, and programs performing support utilities. There was also a control program bundled in the package, which provided the necessary interface between the applications and the computer itself. This was called the Airline Control Program (ACP).

ACP and PARS met the required performance criteria by providing limited functionality and an architecture based on fixed record sizes and requiring programs to be written in assembler language. There were two record sizes supported for disk files, 381 and 1055 bytes, and a third for main storage use of 128 bytes. This fixed record architecture prevented storage fragmentation and eliminated the requirement for any sophisticated storage management routines. Performance also was enhanced by not editing the application requests to ACP for validity in the on-line environment. Most operating systems check the format and content of parameters provided by an application program. This takes time and is repeated for each request. In ACP, it was assumed that the program was working correctly and that these interfaces had been thoroughly checked out in the testing phase. Hence the editing function was dropped and the machine cycles, which would have been used for editing, were used for performing the requested function.

PARS had not been installed at the first customer location before it became apparent that other businesses had transaction processing needs similar to the airlines and could benefit from the processing speed and functionality of a control program like ACP. Household Finance proceeded to develop an on-line consumer finance system based on ACP. Before long, other businesses such as the car rental, law enforcement, hotel, and credit authorization industries recognized the advantages of ACP and developed their own applications using ACP as the on-line operating system. United Airlines pro-
duced such a system for its WESTIN hotels using PARS as a starting point. They took the basic functions of PARS and modified them to support the hotel requirements. For example, a passenger name record in PARS became a guest name record in this system. This WESTIN reservations system, called WESTRON, is now the basis for the reservations systems used by the larger hotel chains.

Since ACP was being used increasingly in other than the airlines industry, IBM changed the name of ACP to ACP/TPF, where TPF stood for transaction processing facility, and began charging for ACP/TPF. It had been available free of charge to IBM customers, but the product is now marketed in all industries where transaction processing is a requirement.

**EVOLUTION OF COSTS**

Until 1970, computer equipment was extremely costly relative to personnel cost. The emphasis was on writing efficient code to get the most performance from the smallest machine. Programmers would look for ways to save a byte here, a machine cycle there; and as bigger, faster, and cheaper computers became available, programmers became slower, less efficient, and more expensive. The supply and demand process was taking hold.

The design of ACP usually required the customer to purchase two CPUs; one was used to operate ACP, the other was a backup CPU to be used in case the first failed. When everything was functioning normally, the backup was used to provide batch functions supporting the reservations system for management reporting, program development and testing, and certain utility functions. The batch system did not use ACP, and therefore required another organization. Two teams were formed for each function: an operations staff for controlling the computer and a programming staff for the ACP; a duplicate team was also formed for the backup function.

Today the cost emphasis is almost totally reversed from what it was originally. Computer systems which can handle 50 messages per second cost almost a tenth of what they did 15 years ago. Some of the design constraints placed on a programmer have been lifted or relaxed, and the current hardware architecture and speed theoretically make the programmer’s job easier.

ACP, unfortunately, has changed little in the manner in which it operates. There are still at least two CPUs required as well as the two supporting sets of personnel. The “user friendliness” of the newer software systems (database management, interactive program development and testing, etc.) has not been added to ACP. As a result, the cost of personnel to support an ACP system is 10% to 25% higher than for other systems. Part of the reason for the lack of change is the huge investment that companies have in their current systems coupled with the original design rules of ACP programming concerning re-entry, linkage conventions, record sizes, etc. Most changes which could be made to the ACP system to take advantage of the newer hardware features would require significant changes to the applications environment. These application changes would have to be performed independently by each ACP user, and IBM is reluctant to permit any user to rewrite their applications.

**SUGGESTIONS FOR COST REDUCTION**

ACP has had a long history of success in the transaction processing industry and has shown many techniques, such as the use of fixed record sizes, to be valid. The current struggle for performance gains as well as cost reductions point out that ACP is not the perfect solution. From a software point of view, programmers need more tools to reduce the software development time. From a hardware point of view, the standard IBM approach of combining CPUs, either loosely or tightly, does not give linear performance improvements as seen with the airlines and their multiple CPUs in ACP systems.

**Hardware Changes**

A tightly coupled system or multiprocessing facility provides for the interconnection of CPUs, via a common main storage, in order to enhance system availability and to share data and resources. A loosely coupled system does not have a common main storage but does have the ability to share data external to the CPUs.

A tightly coupled rather than stand alone or loosely coupled system provides a number of benefits. A fault-tolerant system, tightly coupled, is even better. The Synapse N + 1 system has an architecture which is both tightly coupled and fault tolerant. Some of the benefits of this feature combination are as follows:

1. In contrast to ACP, there is only one system to operate, and hence the size of the operations staff is reduced.
2. There is only one operating system to maintain. This reduces the size of the system staff as well as the training requirements of the applications staff.
3. With a tightly coupled, fault tolerant system, single component failures don’t necessarily affect the availability or reliability of the system.
4. Increased capacity can be provided at a much smaller increment than adding a new system module (loosely coupled) or upgrading to a faster CPU (stand alone).
5. The Synapse N + 1 system can have equipment added or removed on-line without disruption to the system. When added, the increased capability is utilized immediately. Removal of equipment does not affect transactions which are in progress as far as the end user is concerned.

**Software Cost Reductions**

Any transaction processing system will experience numerous changes in its lifetime. The software used in its construction should be structured for ease of development and maintenance. This type of software also reduces the development time for most applications. Most structured high level languages, however, are too inefficient in practice (relative to
assembler language) to be used in a real-time environment. There currently are two languages that can provide the needed structure and performance: C language and assembler language when combined with structured programming macros and techniques.

Significant improvements in programmer productivity can be had if the system provides a good database manager that also provides speed and flexibility. The programmer is free of database design and integrity concerns and can concentrate on mainline coding. The combination of function and performance is difficult to find in one product, but there are improvements occurring regularly in the industry. Relational databases offer that function. Many of the implementations, however, are not yet up to the performance requirements.

Network/terminal independence is a must for the future growth and change of any system. More and more tools are appearing on the market to enhance this area, but true terminal independence is a long way off. Until then, frequent re-coding for new devices will be done.

The potential for a programmer to do one-step programming and documenting is needed. A facility similar to the WEB system of structured documentation, developed by Donald E. Knuth of Stanford University for PASCAL, is needed for other languages. This system provides the ability for the programmer to document the program while coding it. The document portion then is split off from the program by a source analyzer (a computer program) for input into a text processor to produce a final document. The remaining statements are fed to the PASCAL compiler. The programmer maintains only one source file, the tedium of documentation is reduced and the quality of the documentation is improved as well as its ease of maintenance.

SUMMARY

Transaction processing systems have come a long way since the fifties. ACP still is the premier system in the field as far as performance and capacity are concerned. The economics of operating systems today, however, have changed, and ACP has not adjusted accordingly. The marketplace is growing; the demand is there, and new systems are being developed with today's costs and performance factors in mind.