To: Distribution

From: T. H. Van Vleck

Date: March 22, 1977

Subject: Multics Transaction Processing

INTRODUCTION

Transaction processing is what most real-world customers do with on-line computers. Airline reservations, insurance claims and policies, medical records, and many other types of data bases are managed by transaction processing environments. Although Multics is ideal for the support of such applications, we currently lack a unified system which makes such support easy. This memo describes our plans for a transaction processing environment.

Transaction Processing

Most manufacturers have at least one transaction processing subsystem available within their operating system. All such transaction processing subsystems follow the same general pattern:

- An operator at a terminal inputs messages
- The messages cause an application program (usually COBOL) to be run
- The application program usually accesses and may modify a big data base
- The application program produces output which is usually sent back to the operator

This sounds something like classical time-sharing, except that commands aren't allowed to read input from the typewriter. However, because the mission of the transaction processing system is quite different from that of a general time-sharing system, transaction processing systems tend to optimize a different set of characteristics.

RESPONSE

The demand for system response in transaction processing systems arises from a need to make operator productivity as high as possible. In some applications, the organization's image may be affected if clients of the organization must wait while a transaction processing inquiry proceeds. Since transactions tend to be of comparable size, scheduling for transaction processing must solve a different problem than that of general time-sharing scheduling. In particular, consistent response may be preferred to optimum response in some cases.

DATA BASE PROTECTION

Since operators are unskilled, and since the data bases being manipulated are complex in structure and of high value, transaction processing systems take special care to insure data base consistency if a transaction program terminates abnormally. A fault in a single program causes any changes it has made to be undone; recovery after a system crash backs out all half-completed transactions' changes to the data base.

MESSAGE HANDLING

Since it may be costly to repeat or omit a transaction which was in progress at the time of a crash, transaction processing systems are also designed to perform careful message bookkeeping so that an operator can determine whether the system has actually heard a request, and whether the request has completed.

Transactions are often input from video terminals, in order to eliminate paper handling and because these terminals are inexpensive. Video terminals are often used in block transfer mode, with protected fields defining forms to be filled in; a single transaction message may therefore be quite long.

Because operator training is costly, transaction processing systems provide elaborate facilities for tailoring the operator interface.

THROUGHPUT

System efficiency is less important than the functional issues described above. Efficiency cannot be gained at the expense of reliability. Nevertheless, it is important to be able to obtain very high throughput with almost all system features in use, including the data base manager, journalization, and COBOL.

Since many transaction processing systems spend 80% of their time in system code, the natural efficiency of Multics gives us a

good chance to be cost competitive. (Many other operating systems cannot take advantage of multiple CPU's, for example.) Some orders of magnitude are in order:

USGS benchmark	14 t/sec
Alcoa	1
IRS benchmark	30
Marketing wish	100

Because the transaction processing environment is well defined, not all the facilities of a fully general user program need to be provided for each application program's environment.

TRANSACTION PROCESSING ON MULTICS

This section describes our plans for Multics transaction processing. Actual implementation may be broken into several phases.

Interfaces

The Multics transaction processing implementation supplies three important interfaces: the administrator interface, the application programmer interface, and the operator interface.

ADMINISTRATOR INTERFACE

The administrator of a transaction processing subsystem on Multics will use table compilers to specify the following items:

- Lines and terminals

  Dialed versus dedicated; particular operator; group membership; authorization; signon required; ID code required.

- Data Bases

  Journalization; locking; audit trail; opening.

- Application Programs

  Permission required; keyword mapping; conversational; maximum CPU limit; priority; termination on deadlock; simultaneity; error handling.

- Operators

  Password; privileges; group membership.

The transaction processing administrator will compile these tables and will be able to install them while transaction processing is running.

The transaction processing administrator must also function as a data base administrator for his data bases. This involves him in both definition and security considerations.

To start transaction processing, the administrator executes the command

start_tp

in a process which then becomes the transaction processing master process. During startup, the master process will discover if data base recovery is necessary, and if so will operate on the data base and the journal file to produce a consistent situation. Transactions which were incomplete at the time of the failure will be automatically restarted, unless the administrator has specified otherwise.

The master process may be started automatically by the system_start_up.ec on every bootload if it is a permanent part of the system's capabilities; or a permanent absentee job can be created to always start the application.

APPLICATION PROGRAMMER INTERFACE

Application programs to be run under Multics transaction processing must obey a few simple rules. They may be written in any language: COBOL, PL/I, FORTRAN, BASIC, APL, LINUS and even ALM will work, as will user-supplied compilers and procedures specified in Multics command language via exec_com. Application programs will obtain their input in a standard fashion. The programs must not attempt to read from the terminal; attempts to read or ask questions will terminate the program. The application programs must not use data files other than those defined by the transaction processing administrator if they wish to have full failure backout protection.

The Multics Data Base Manager, with both MIDS and MRDS interfaces, will probably be heavily used in this environment. Special options to avoid redundant opening and closing of files, both MDBM and regular language I/O, will be provided.

OPERATOR INTERFACE

The operator interface to Multics transaction processing can be tailored to meet the needs of the application. The transaction processing administrator defines what the commands are, how the operator signs on, and how messages will be routed. The standard Multics terminal software provides device-independent support for all terminals.

Structure

The transaction processing master process, defined above, causes several other processes to be created to perform the actual transaction processing.

One or several message control processes will be created to handle terminal input/output and message queueing. The message control processes manage groups of terminal lines; they may perform demultiplexing on multidrop lines. The lines handled may be dedicated or attached to the control process by the dial facility.

One or several worker processes will be created to perform the actual application execution. The worker processes will execute one transaction at a time, queueing output messages which will be sent to the operator if the transaction completes correctly. The worker processes establish an environment which protects against abnormal termination of an application program. The start of each transaction causes the worker process to make a mark in the recovery journal file so that cleanup operations can discover what operations were in progress but had not completed at the time of a crash or fault.

The worker processes cooperate with the data base software and the transaction processing master process to protect the subsystem from deadlock.

Batch processes may also operate on the same data bases which are in use by a transaction processing application; batch processes also are protected from deadlock.

Figure 1 shows a simplified diagram of the relationships of the various processes.

Other Features

Many other features are planned for the Multics transaction processing subsystem. Some of these will be present in the initial version, and some will be deferred until we are sure of the most general way of meeting customer needs. A list of such topics follows:

sophisticated scheduling
forms terminal support
message routing
testing interface
training interface
keeping multi-line messages together
batched transactions
inter-terminal messages
broadcast messages
starting absentee job from transaction
transaction chaining
updating programs without shutdown
statistics
AIM multilevel security