J F OSSANNA L E MIKUS and S D DUNTEN
Communications and input-output switching in a multiplex computing system
FJCC page 231 1965
W W LICHTENBERGER and M W PIRTLE
A facility for experimentaton in man-machine interaction
FJCC page 589 1965
J W FORGIE
A time- and memory-sharing executive program for quick-response on-line applications
FJCC page 599 1965
J D McCULLOUGH K H SPEIERMAN and F W ZURCHER
A design for a multiple user multiprocessing system
FJCC page 611 1965
W T COMFORT
A computing system design for user service
FJCC page 619 1965
B W ARDEN B A GALLER T C O'BRIEN and F H WESTERVELT
Program and addressing structure in a time-sharing environment
J of ACM no 1 page 1 1966
J B DENNIS and E C VAN HORN
Programming semantics for multiprogrammed computations
C of ACM no 3 page 143 1966
G F LEONARD and J R GOODROE
More on extensible machines
C of ACM no 3 page 183 1966
J M WILLIAMS
Design of the real-time executive of the UNIVAC 418 system
ACM National Conference page 401 1966
M J MENDELSON and A WENGLAND
The SDS SIGMA 7: A real-time time-sharing computer
FJCC page 51 1966
R J GOUNTANIS and N L VISS
A method of processor selection for interrupt handling in a multiprocessor system
Proceedings of IEEE no 12 page 1812 1966

Systems recovery from main frame errors

by R. ARMSTRONG, H. CONRAD, P. FERRAIOLO, and P. WEBB
Burroughs Corporation
Paoli, Pennsylvania

INTRODUCTION

"If you look at Automata which have been built by men or exist in nature, you will very frequently notice that their structure is controlled only partly by rigorous requirements and is controlled to a much larger extent by the manner in which they might fail and by the (more or less effective) precautionary measures which have been taken against their failure. There can be no question of eliminating failures or of completely paralyzing the effects of failures. *All we can try to do is to arrange an automaton so that in the vast majority of failures, it can continue to operate.*"[1]

The operational system built by Burroughs Corporation is a communications automaton. It uses conventional computers, conventional computer peripherals, equipment for tying these computers into a communications network, and software for the operation, scheduling, and automatic recovery from failures within both the hardware and software systems.

The significant contribution of this combined hardware and software system is that it provides automatic operation and automatic recovery from errors without duplicating its operations. New electronic telephone systems, for example, provide automatic recovery by duplicating operations and comparing results. This new communications automaton developed by Burroughs Corporation may provide automatic recovery by having only one spare module of each major black box of the system; automatic recovery is achieved by software manipulation of the operating hardware judiciously substituting these spares under program control and keeping track of the performance of each black box operating in conjunction with the other black boxes in the system. Black box modules eliminated from the operating system by the software are repaired off-line by the servicemen.

The reason for developing this new automaton is to provide an ultra-reliable system which is required to give service twenty-four hours a day, seven days a week. The applications for such systems are typified by command and control systems with a much wider sphere of applications possible in the future.

The system designed to meet these stringent requirements is a modular one comprising carefully integrated hardware and software. A sufficient number of modules of each type, computer, memory, I/O controls and peripherals are selected to perform the required function. This configuration is termed the operational system. Then a second system is composed of at least one module of each critical element. A critical element is one whose loss would disable the system. This second system is called the back-up system.

Whenever possible, the two systems run in parallel under the supervision of the automatic recovery program. The operational system performs all required functions and monitors the back-up system. The back-up system constantly repeats a series of diagnostic tests on the computer, memory and other modules available to it and monitors the operational system. These tests are designed to maintain a high level of confidence in these modules so that should a respective counterpart in the operational system fail, the back-up unit can be safely substituted. The back-up system also has the capability of receiving instructions to perform tests on any of its elements and to execute these tests while continuing to monitor the operational system to confirm that the operational system has not hung up.

When the system is running in this duplex mode, failures can occur in either of the two subsystems. The occurrence of a failure in either of the two subsystems initiates the error recovery cycle.

During normal operation, the operating subsystem is running under the control of the operating and scheduling software, and the backup subsystem is operating under the control of the diagnostic software. When a failure is detected anywhere in either system, software control of all hardware is turned over to the diagnostics, and operations are suspended.

This phase of activity is called the error recovery cycle.

The error recovery cycle consists of five phases: error detection, error isolation, reconfiguration, data recovery and resumption of normal operations.

Error detection

The error detection phase relies on the following capabilities and procedures.

Each program is capable of detecting failures in it own equipment.

Each of the program systems has the capability of signalling the other system in the event a failure is detected in the critical modules under its control. The program system receiving the communication will initiate the error isolation phase of the recovery cycle.

Each program system maintains a real-time count in a memory module and periodically updates this count, which indicates the operating status of the system. Each program system periodically checks the real-time count of the other system; this check is made to verify that the count has been updated. If the real-time count is equal to a previously checked value, the error isolation phase of the diagnostic program is initiated. This feature is meant to guard against the few cases in which a computer comes to an unintended stop.

Also, if either system encounters a no-access interrupt while attempting to read from the other system's hardware status table as maintained by the software, the error isolation phase of the diagnostic program is initiated. A no-access interrupt is created whenever access to any system black box cannot be achieved.

In summary, the error isolation phase of the recovery cycle is initiated by either the active system or the back-up system for any of three reasons: (a) the receiving of a communication from the other system to the effect that a failure has been detected in a main frame module of the data processing set, (b) the detecting of a failure in the real-time count of the other system and (c) the receiving of a no-access interrupt while attempting to access one of the other system's memory modules. When the error isolation phase is initiated, the computer that has detected the error is halted, and the *other* computer is used to run error isolation. Thus the computer and memory module normally used to perform error isolation are ones in which confidence has been reasonably established ahead of time. Exception to this rule will exist if the active system attempts to run error isolation on itself in the unlikely event that the back-up system is inoperative due to a previous failure.

A further exception, which also is in the spirit of John von Newman's finite limit to failure protection, is the case when a failure in one machine is of such a nature that it points to the other machine as having failed, when in reality the other machine is good. Such a failure has not been detected in our extended operating experience with this system; if it occurred, the system would go down.

Error isolation

When the error isolation mode is initiated, the diagnostic program becomes owner of all available equipment, and the diagnostic test programs required for locating a fault are executed.

Since all information in the operating system is suspect, once an error is discovered, the diagnostic programs initially destroy all information and load themselves from the one, or preferably replicated duplicate sources on which the programs are stored.

In the checking of the hardware modules, all available magnetic drum storage units are used. Specifically, the test programs executed during error isolation are the computer, switching interlock, memory, and I/O test programs. Following the execution of error isolation, one of the main frame modules will always be updated so that its status is bad. If no errors are detected during the error isolation cycle, the halted computer which detected the error, is assumed to be at fault and is indicated as the faulty module. If an error is detected, an error message identifying the faulty module is constructed and printed on the high speed printer.

Reconfiguration phase

The reconfiguration phase of the error recovery cycle is not executed at a singse point in the cycle but rather at several discrete points. The basis of reconfiguration is the system status table. During the error isolation phase, this table is adjusted to reflect the status of each of the elements based upon the results of the test run at that time. If a module is faulty and marked so in the status table, no attempt to use it is made and the rest of the recovery cycle is relatively easy. If a memory is faulty, the reload program must take this into account and at least some of the program system must be reloaded into areas different from those used formerly.

Data recovery phase

The last act performed in the error isolation phase causes the operational executive program and the data

recovery program and certain fixed tables to be reloaded into the system. In the normal error recovery case, the primary function of the data recovery program is to restore the variable information that existed in memory just before the error which caused the recovery cycle occurred.

The final action of the data recovery phase consists in loading the rest of the operational program system over itself causing itself to disappear slowly and in piecemeal fashion, somewhat like the Cheshire Cat, and normal processing to resume.

Resumption of normal operations

The last function of the data recovery phase causes normal operation to resume.

CRITIQUE

For our original design, the time required to execute the recovery phase just described is in the order of two to four minutes contingent upon the type of failure and the traffic load at the time of failure. During this period of time service is disrupted.

A critical evaluation of that original system revealed that significant improvement could be made by the addition of some particular error detection circuitry. In the original system it was frequently impossible to determine the precise module that failed without running extensive tests using different combinations of equipment. If computer A ran well with all memory modules but one, it was not necessarily true that that memory module was at fault. It was necessary to repeat the test with computer B and the suspect memory. The effect was the diagnostic program required the use of the total system to isolate a failure in a single element.

The addition of some error detection and test circuitry (specifically adding memory parity checking) has made it possible to isolate failed modules in a matter of milliseconds where previously seconds were required. In addition, in many instances it is no longer required to interrupt the operational system to perform the isolation and recovery process.

Further analysis has revealed that, were we to redesign the system or develop a similar system at this time, it would be possible to provide isolation of failed modules in a matter of microseconds in ninety-five percent of the cases where seconds are otherwise required without incurring great expense.

To accomplish this it is necessary merely to recognize as a legitimate goal that individual modules should be readily recognized as faulty when they are, and to provide the requisite facilities to make this determination.

In summary, the significance of what we have done is creation of an automaton which operates and recovers itself automatically. A different system concept is used which saves duplicating the entire set of operating hardware by substituting software controlled testing and interchanging of the few spare modules.

The significant information we have learned that is of value for future software and hardware system design, is that adequate hardware checking within each black box module is necessary such that the source of the errors it indicates are unambiguously reported to the software. We needed more clarity in differentiating between errors occurring in the I/O controls from errors occurring in the I/O peripherals themselves.

The significant accomplishment is that we reached our goal: "that in the vast majority of failures... (the system) can continue to operate."

REFERENCES

1 J VON NEUMAN
The theory of self-producing Automata
1966