IEEE Transactions on Nuclear Science, Vol.NS-22, No.3, June 1975

CONTROL THROUGH A SYSTEM OF SMALL COMPUTERS

K. B. Mallory
Stanford Linear Accelerator Center
Stanford University, Stanford, California 94305

Introduction

It is still possible to operate SLAC's linac without any computers. Normal operation, however, now relies heavily on the system of eleven computers varying in sizes from 4K to 16K words of memory. By using a large number of small computers, the I/O demand on each computer is kept small. The use of disks and a programming system that exploits multiprogramming and program overlays allows the computers to execute large numbers of tasks in a virtual memory several times their real size. The response times demanded are moderate, but the multiple input terminals, the different types of links between computers and the different sizes of the computers require a continuing traffic analysis to reduce the probability of "rush hour peaks" jamming too many tasks into any one computer.

Status of System

The original purpose of SLAC's SDS-925 in the switchyard control room and of the PDP-9 in the accelerator control room was to perform routine chores automatically as an aid to the operator. In 1971 this goal was changed to provide a method to operate accelerator and switchyard from a single control room.[1] The method chosen was to link the two computers, to finish connecting all accelerator signals to the PDP-9 and to emulate the existing manual control functions on touch panel display units at the new Main Control Center.[2] One major improvement was a more flexible trigger-defining system for setting up beams, better correlation of multi-level devices with the beams they control and better human engineering of the setup controls.[3]

As a result of consolidating the two control rooms, however, it became no longer possible to display so much status information about the accelerator at one time; operators could only adjust one control at a time; and it became difficult to obtain and display some of the analog signals. During the past two years, a set of panel displays has been devised that provides new types of status summary to compensate for the limited display area. Nine PDP-8's have been installed to improve access to accelerator signals and controls.[4] The operators now have more control parallelism than they had under the original manual control system, and an improved analog acquisition system will be completed this summer.

It is now time to return emphasis to convenience features for the operators. The original magnet set programs for the BSY were removed when the SDS-925 was reprogrammed to handle the touch panels. They must now be restored. A program in the PDP-9 to record and later restore sets of quadrupole settings in the accelerator has never been made available from the touch panels. Beam diagnostics programs, while of lower importance in a linear accelerator than in a circular machine, should be provided. These are all software additions, since the required signals presumably have all been interfaced to the computer system. We expect that many new programs can be added without degrading system performance because of the unique features of the executive program used in all of the computers.

Executive Program

The executive programs in the SDS-925, the PDP-9 and the PDP-8's are all versions of the same Disk System (DS) executive.[5] DS provides for multiprogramming a large number of tasks (typically up to 25 in a PDP-8, well over 100 in the SDS-925) and multiprocessing, in that a task may be transferred from one computer to another for continued execution. Tasks may be initiated by the system in response to I/O input or by other tasks. Supervisor calls are provided for real-time waits and for communication between tasks and the system by means of Events. (A task may wait for an event to be declared by the system or by another task. A number of parameters may be passed to the task, if desired. The program declaring the event is informed if a task was released by the event.) The major paths of task flow and communication are indicated in the figure on the next page.

The primary concept of DS is that a "task" consists of a list of arguments obtained from a "free list" established when the system is loaded. These arguments include information for scheduling the task, a page name and entry-point location pointing to the code to be executed, and a variable number of parameters that are saved by the program during supervisor calls. A secondary concept is that all code should be reenterable after each supervisor call, so that many tasks may share the same code and so that a task will execute properly if the return from a supervisory call finds itself in a new copy of the code, possible in a different part of core. Parameters of a task may therefore not be stored on the program page during a supervisor call. DS provides system subroutines to allow a user to locate, create and delete arguments as required.

A queue consists of a list of arguments which are pointers to tasks. Two queues are maintained by DS: an active queue which contains tasks due for scheduling in past or future time and a blocked queue containing tasks waiting for named events. A minimum of five arguments must be obtained from the free list when a task is created. A task may obtain more arguments from the free list when it requires them (or return unused arguments) during execution. When a task is terminated, all of its remaining arguments are restored to the free list. The number of tasks that can be handled by the system is limited by arglist overflow; that is, a situation in which the free list is empty and additional arguments are not available when required.

If there were no error-recovery, one would have to halt and restart the system when arglist overflow occurs. A first step of error-recovery is to purge the task which requires another argument when the free list is empty. This can result in losing an essential task like the keyboard or link handler. It sometimes purges one of a pair of co-tasks leaving the other hung forever. In a PDP-8, overflow can occur in the foreground, and it is not clear how to purge the partial task that is involved.

Normally, tasks are terminated and their arguments returned to the free list at a rate faster than they are created. But occasional bursts of new tasks or a concentration of tasks waiting for the same resource can produce peaks, which deplete the free list. By reducing the intensity and frequency of these peaks, the system may be made to operate more smoothly and reliably.

The installation of the PDP-8's was required primarily to reduce contention for the single control channel connected to the PDP-9 and for its single ADC multiplexer. The PDP-8's have provided 31 smaller ADC multi-

*The work described in this paper was supported by the U. S. Energy Research & Development Administration

plexers and 33 parallel output control channels at considerably lower cost than adding the equivalent resources to the PDP-9.

Traffic Study

We maintain a continuing traffic study of the flow of tasks within and between CPU's and of the utilization of resources such as the disk, links, "real-world" interfaces and internal I/O buffers. We have identified a number of situations where contention may be reduced, the rate of creation of tasks may be lowered or a task may be brought to an early termination. This sometimes reduces the total work accomplished, but has resulted in large gains in overall system efficiency.

Contention for disk access is a major cause of queuing of tasks. DS reduces this contention by never rolling a program out and relieves "thrashing" by overlaying only the program page that has been idle longest and only then if the page has been in core long enough to have been used at least once. We have also found it desirable to write frequently used code into a single program page to reduce the number of different pages required.

The following techniques (with examples) have been found useful in limiting the rate of creation of tasks.

1. Change mode of operation of program
   The status monitoring program normally initiates a separate task to update operator displays for each status change in the accelerator. When more than 20 changes are detected in a frame of the status multiplexer, the program stops reporting individual changes and sends instead an update of all accelerator status in a single message.
2. Force tasks to be serial instead of parallel
   If a value must be updated on several of the touch-panel displays, a single program performs each update in sequence, instead of initiating a separate task for each panel.
3. Suppress task initiation
   In the PDP-9 and the 925, link messages are not initiated if the free list is too short. (This has the disadvantage of dropping the newest data in favor of older and possibly obsolete data.)

We have adopted a number of methods to reduce contention for other resources.

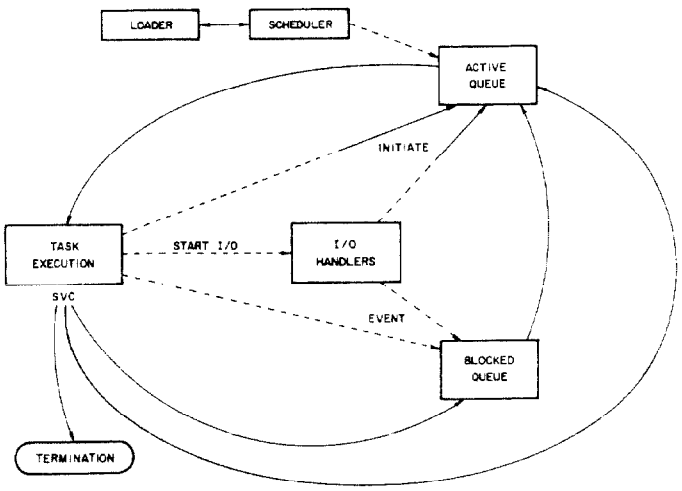
1. Eliminate duplicate tasks
   When requests for analog values arrive at the PDP-9 faster than the values can be returned to the 925, the elder requests are terminated.
2. Terminate tasks before free list is depleted
   In a PDP-8, when the free list is short, the eldest task waiting for a link is terminated if there are more tasks waiting for that link. (This only occurs, however, when the resource is released and there is a chance to check if more tasks are waiting for it. It has the advantage that the eldest tasks are terminated and the queues are kept current, but it can do nothing about tasks queuing up during a single "busy" period of the resource.)
3. Reject conflicting tasks
   When the PDP-8 is busy with a continuous "adjust" command, commands to adjust a different parameter are terminated instead of queuing up, since they will continue to be sent until the first command is complete and the second is then allowed to produce the required effect.
4. Combine related tasks
   When several messages for display are created in quick sequence, the separate messages are given to a single task for rewriting the display.

Conclusion

Each of the computers in the control system is idle much of the time. By controlling the peak traffic, we expect to be able to add many "occasional" tasks to increase the aid the computers can provide to the operators.

References

1. a) K. Breymayer et al, "SLAC Control Room Consolidation Using Linked Computers," SLAC-PUB-866, March 1971.
   b) S. Howry et al, "SLAC Control Room Consolidation -- Software Aspects," SLAC-PUB-871, March 1971.
2. W. Struven, "Experience with Touch Panel Control at SLAC," SLAC-PUB-1191(A), March 1973.
3. S. Howry, "Trigger Pattern Generation," SLAC-TN-75-5, March 1975.
4. W. Struven, "SLAC8's - A Distributed Accelerator Control and Monitoring System," this conference.
5. a. S. Howry, "A Multiprogramming System for Process Control," SLAC-PUB-949 (MISC), August 1971.
   b. S. Howry, "DS: A Virtual Memory Executive Program for the SLAC Control Computer," SLAC TN-75-4, March 1975.
   c. K. Mallory, "Sam Howry's DS Executive Translated for a PDP-8," SLAC-TN-75-6, March 1975.



TASK FLOW AND COMMUNICATION UNDER DS