EXPERIENCE IN USING WORKSTATIONS AS HOSTS IN AN ACCELERATOR CONTROL ENVIRONMENT*

A. Abola, R. Casella, T. Clifford, L. Hoff, R. Katz, S. Kennell, S. Mandell, E. McBreen, D.P. Weygand
AGS Department, Brookhaven National Laboratory
Associated Universities, Inc., Upton, New York 11973 USA

Summary

A new control system has been used for light ion acceleration at the Alternating Gradient Synchrotron (AGS). The control system uses Apollo workstations in the dual role of console hardware computer and controls system host. It has been found that having a powerful dedicated CPU with a demand paging virtual memory OS featuring strong interprocess communication, mapped memory shared files, shared code, and multi-window capabilities, allows us to provide an efficient operation environment in which users may view and manage several control processes simultaneously. The same features which make workstations good console computers also provide an outstanding platform for code development. The software for the system, consisting of about 30K lines of "C" code, was developed on schedule, ready for light ion commissioning. System development is continuing with work being done on applications programs.

Introduction

For the past two and one-half years we have been building a new control system for the Brookhaven AGS. This control system combines the roles of console computer and system host through a network of workstations. A workstation is a single user, 32-bit microcomputer with a high resolution, bit mapped display; a mouse pointing device; and several megabytes of memory. It runs a multitasking operating system and is networked to other code-compatible nodes which share a common file system. The workstation display supports windowing so that the output of many processes may be displayed simultaneously.

The console/host is one component of a distributed control system which has three functional layers.[1] It is the top-most layer furthest from the accelerator devices and is not a part of the realtime control of the accelerator devices. The new control system has been in use for controlling all the equipment of the transfer line from the Tandem Van de Graaff to the AGS, and some of the equipment in the AGS used for light ion acceleration. Our experience in building and operating this system has convinced us that workstations, serving the dual roles of hosts and console computers, create a suitable environment for the development and operation of control systems. A look at controls activity at other laboratories indicates that workstations will be playing an important role in many future accelerator control systems.

The next sections of this paper will present a short history of the project accompanied by our observations on using workstations for code development followed by our observations on the advantages of workstation-based consoles.

*Work performed under the auspices of the U.S. Department of Energy.

History

In early 1984, the Accelerator Controls Section (ACS) began to consider workstations as possible operator console/hosts for future accelerator systems at the Brookhaven AGS. One of the attractions of a workstation-based system was the modest initial cost of a workstation and the easy incremental growth options available as the system expands. The modularity of a workstation-based system offered the opportunity for continued low cost evolutionary modernization of hardware. The large bit mapped screen and mouse seemed to be a good operator interface for a control console thus negating the need for special hardware or software development.

One of the main arguments against the workstation-based control system was capacity. Would a system without the traditional large time shared computer as a host be able to provide enough access for developers and other users, who seem to roost on controls system computers. Secondly, there was also some question as to whether the microcomputers would be powerful enough to serve as host to the control programs needed in running an accelerator.

Selection of a workstation in early 1984 was easier than it would be today. Then there were only two vendors. We took delivery of our first Apollo Domain Node, a DN460, in September of 1984.

Almost simultaneously with the ordering of our first system was the decision by the ACS to propose the use of workstation-based consoles for the control of the ion transfer line. This line would bring ions from the Tandem Van de Graaff to the AGS. The control system would be needed by January, 1986.

Early decisions in the development of the new consoles were to strongly influence our development efforts on the Apollos. First, we would concentrate on general control tools upon which a control system would be built. Controls tools would be written in "C". Second, we would not hide the Aegis operating system under a layer of generalized OS calls, but would deal with it directly.

Our initial experiences in developing the system were mixed. On the positive side were the general tools to provide for program development, interprocess communication and access to multibus, our connection to the outside world. With no experience in either the Aegis, "C", or the commercial IEEE-488 multibus board, we wrote a driver for our local data network working in less than three man-months.

Problems did occur. There was almost immediate recognition that a workstation was a single-person device. Time shared access to the workstation was not suitable for program development. Every programmer needed a node. Building our operator interface tools, a good menu program, was much harder than

CH2387-9/87/0000-0594 $1.00 © IEEE

expected. There was a level of software missing that would have allowed us easy access to menu-type objects. Manipulating bit mapped display proved more complicated than simple memory mapped alphanumeric displays. Our choice of "C" as our language conflicted with the Pascal written OS causing our calls from "C" to be hard to read. We discovered that we could pass arguments to OS routines that would cause problems in the OS because the Pascal argument was lost.

By January of 1985 the basic design of the entire system had been completed.

The console would contain two Apollo nodes, one of which would provide access to the control network and drive the basic device control program. This general device control program would not use the bit mapped display, but rather the more traditional character memory mapped video screen. This would keep the Apollo screen available for applications programs. At this time we were also given a May 1985 date as a milestone for having a demonstration of the basic device control program.

The first level of general programs that must be operational for the January, 1986, commissioning would be:

1. Device list access program.
2. Device control program.
3. Alarm displays.
4. Save/restore of device commands and set-points.
5. An applications program for selecting analog signals.
6. A device database.

The May milestone was met; although the system was far from operational, we were able to control and view devices and to display alarms on an alarm screen. We had generated 15K lines of "C" code and had a database management system application for managing the device database.

The months from June, 1985 to January, 1986, were spent in moving from demonstration programs into production code. The control consoles and many of the devices were commissioned along with the transfer line itself between January and March of 1986.

By this time, with a staff of five programmers (see Table I), and over a year's experience with the system, we had begun to appreciate and make use of the many features of our networked workstations and were making good use of the development tools of our system.

Table I

	number of nodes	lines of code	program staff	total man months	comments
Sept 84	1	0	2.0	0	Begin
Jan 85	2		2.0	8	Design done
May 85	5	15k	3.0	18	Demo in place
Sept 85	5		4.0	30	Console installed
Jan 86	7	30k	5.0	46	Commissioning Application start
May 86	7		5.5	66	
Sept 86	7		5.5	89	
Jan 87	10	50k	5.5	110	

Because each developer has a dedicated node, no special time on the system needs to be allocated for low level development or debugging. Each node becomes a mini-console for development. Device drivers, servers, and clients can be under development on one node while production versions run on other nodes in the network. Intertask communication becomes easy when two communicating tasks can be run in separate windows on the same screen, both in the source code debugger.

The quick access to program files, a point and a click with a mouse, along with the extensive use of a multiwindow source code debugger, makes source code public. Because it is so easy to read other programmers code, there is pressure on programmers to write code that can be easily understood by others. This seemingly minor feature of the workstation environment is of large importance in the process of code development. Listings become a thing of the past, editors search code rather than programmers and the system under development becomes much easier to learn. Early on in the development cycle we began using an interactive source library system to control access to the source code. This allowed multi-developmental paths to be pursued without effecting other developmental or operational programs.

Wherever it was possible, we made use of features and programs already in existence for building the control system. All our interprocess communication and queuing of messages makes use of operating system services. A commerical database, SIR, was used for our device database. The database is converted into a large "C" structure which is made available for reading by programs via the mapped memory feature of the operating system. The operating system directory/file tree structure is used by the operator's tree program to provide access to the devices in the accelerator. The program merely moves up and down through directories which mirror the structure of the devices in the accelerator.

In the year since that first run, the system has continued to grow. The basic system is getting stronger. Higher level applications and monitoring programs are being written and put into production.

Observations

Most of our initial assumptions about workstation based consoles have been verified by our work with these consoles over the past year. In normal operation, the console screen displays many icons (~ 10), which represent running programs that the operator can view and interact with by just clicking the mouse button. This one screen and mouse replace three or more screens and some select buttons on the old consoles in the AGS control room.

We have already upgraded our console computer from the first model which was available in 1984. We were able to increase our processor power and memory, each by a factor of two, with the 1986 model workstation. Our original configuraton was not powerful enough because of a lack of physical memory and screen refresh speed. The waiting for a process to page into memory in response to a command was annoying as was the slow refresh. The upgrade consisted of moving the original workstation out of the console and placing the new one in its place. This was done in about an hour.

We were wrong in thinking that terminal access to the system would be useful for program development. The multi-window, multi-process, mouse environment is addictive for programmers. None of

our development work was done via terminal access to the workstation. Fortunately prices are now such that a development node is well under $10,000.00, so we have been able to provide nodes for program development. The system is still new so we do not have a community of users outside of the controls section. We expect that as the user community grows, the price of entry level workstations for our network will come down to the price of a Mac or PC.

An operations console needs more than one display device. Even though the large bit mapped multi-window workstation's screen is an extremely powerful display device, there needs to be some dedicated screens for at least alarms and monitors (comfort displays). Screen management on the primary screen is a problem. After an initial attempt at designating some areas of the screen for particular functions, we have left the problem of screen management to the operators. They have tools, provided by the system, to change the size, shape and location of the output window of each of the tasks, and can lay out the screen as they wish.

There is also flexibility in the layout and use of the console itself. When the console in the Tandem Van de Graaf was upgraded, the staff there chose to keep both workstations in the console.

Because only the workstation is used as a control console, each of our nodes becomes in effect a mini-console. This has clear advantages for development, debugging and access to the accelerator. It also presents the danger of having too many cooks at work controlling the same devices at the same time.

The success of the first console has convinced us that our view of a control room with many small general purpose consoles is correct. Our first pass at the general console was flawed in that it was oriented too much toward a single user. During normal operation, each console is a single user station but when studies or commissioning is going on, the general purpose console must support more than one user. This second user is likely to be involved with the analog signal display in the console. We feel that we must provide for independent operations of the analog substation of the console. Because the multiplexers in the system are under computer control, the analogue substation needs a display and input of its own.

Acknowledgments

We would like to credit J. Niederer for first pursuing workstations for controls work. We would like to thank the AGS management for giving us the freedom to test our ideas for this new control system. Finally, we owe a big thanks to the operations staff of the Tandem, our first users, who were patient and helpful in the commissioning process.

References

1. A. Stevens, T. Clifford, R. Frankel, Distribution of Computer Functionality for Accelerator Control at the Brookhaven AGS, IEEE Trans. Nucl. Sci., NS-32, 2023 (1985).