Decentralized shared computers in the National Earthquake Hazards Reduction Program

Peter L. Ward, James Herriot, and William F. Jolitz

U. S. Geological Survey Open-File Report 81-104

This report is preliminary and has not been reviewed for conformity with U. S. Geological Survey editorial standards and stratigraphic nomelclature.

Decentralized Shared Computers in the National Earthquake Hazards Reduction Program*

Peter L. Ward, James Herriot, and William F. Jolitz

Office of Earthquake Studies
U.S. Geological Survey
345 Middlefield Road
Menlo Park, California 94025

ABSTRACT

The needs for computer analysis in the National Earthquake Program are varied, complex, and rapidly growing. Major sources of frustration have been the difficulty of keeping up with the required analysis, and the difficulties of sharing a wide variety of data and computer programs among several hundred researchers at dozens of institutions located throughout the U.S. A major effort is underway to create a similar computer environment running on many PDP 11/70 minicomputers located across the country that encourages widespread sharing of data and "software tools" without impeding the free flow of creative ideas in this research program. Large libraries of high-level tools are provided in a well documented manner. The tools can be readily combined into programs using a new interactive language, any compiled language, or, in many cases, the Shell command language of the UNIX** Timesharing System. New tools are easily written by any user. Major system tools provided include advanced general plotting programs and a relational indexed database. Extensions to the operating system include text overlays and an 830,000 baud graphics interface.

Background

The U.S. Geological Survey and the National Science Foundation are responsible for the management of the National Earthquake Hazards Reduction Program. The goal of this program is "to reduce the risks of life and property from future earthquakes in the United States" (Earthquakes Hazards Reduction Act of 1977, 95th Congress). The program is carried out by scientists at several government and industrial laboratories and at dozens of universities across the country (Hamilton, Geological Survey Circular 780, 1978). A major part of the research involves the collection and analysis of data recorded continuously from over 1000 seismographs with data rates of

*Presented on June 17, 1980 at the National USENIX Conference, Newark, Delaware
Open-File Report 81-104. This report is preliminary and has not been reviewed for conformity with U.S. Geological Survey editorial standards. Any use of trade names is for descriptive purposes only and does not imply endorsement by the USGS.
**UNIX is a trademark of Bell Laboratories

100 samples per second, and 500 lower frequency instruments such as tiltmeters, strainmeters, creepmeters, and magnetometers with data rates of from one sample per second to one sample per ten minutes. These instruments are located in networks primarily in California but also spread throughout the United States from New England and Missouri to Alaska and Hawaii. The data are telemetered to local research centers, processed, reported routinely in bulletins and then analyzed by researchers in a wide variety of continuously changing ways. Other types of data are collected when the larger earthquakes occur or in special field studies focussed for a short period on a specific area.

Computational needs

Needs for computers in this program range from real-time, on-line collection and analysis of data, to routine processing of large amounts of data (up to 30 or 40 million bits per earthquake), interactive and non-interactive research analysis, large numeric modeling programs, word processing, and maintenance of financial records. There is a fundamental conflict in these needs: the routine processing of large amounts of data requires high computational efficiency and speed using relatively fixed programs whereas the research processing requires high efficiency in programming since a wide variety of routines are needed and these routines are constantly being developed and changed. The first case is compute intensive whereas the second is person-power intensive.

Trying to share computers, data, and computer programs among a wide variety of institutions is particularly difficult in a research environment. Research scientists by nature and training are independent, creative, and skeptical. There are a wide variety of opinions at any given point on how to procede. Some duplication in a research program is not only desirable but necessary. The techniques that we believe will be most important in five years may in fact not turn out to be useful and yet computer procurements and programming efforts take years to carry out. The major cost in a research effort is in salaries and overhead. Thus a major problem within these kinds of constraints is this: How do you create a computer environment that will encourage widespread sharing of data and programs but will not constrain the free flow of ideas and will respect the different needs and approaches of many first-rate scientists at a wide variety of institutions? The ultimate goal is to use a minimum amount of the limited human energy and money to meet effectively the computational needs of the Earthquake Program.

Alternatives

There are a variety of alternative approaches and combinations of approaches to this problem. We will discuss some of the pros and cons of a few of the alternatives here. From the hardware point of view, we could set up one or two large centralized computer systems that all researchers in the program would use. This alternative was ruled out because of the need for high-speed interactive display of large amounts of time-series data. The cost of telemetry adequate to handle such work would be prohibitive within the scale of a highly labor-intensive 30 million dollar national program. Furthermore the primary need for quick access to the data is at the institution where the data are collected. Scientists at other institutions are typically satisfied to receive such data by mail. Much of the analysis needs to be done on-line as the data are collected and telemetry would need to be 24 hours per day.

Another alternative for hardware is to buy similar minicomputers for each site. This approach has the drawback of having to fit programs

within small machines and requiring staff at each site to operate the machines. Some clear advantages are immediate access to the data locally at high rates of speed, local control, and a distributed processing capability that would likely be needed anyway at least to handle the immediate on-line analysis of network data. This choice was the one adopted. A committee of users was formed to draw up the specifications for the systems. A competitive request for proposals was issued and six systems were bought along with the option to buy many more.

From the software point of view, there were also a variety of alternatives. At a minimum some standardization of data format is required. This format must, in part, be binary because of the large volume of time series data. Furthermore, the format must be versatile enough to handle a wide variety of time-series and tabular data. The format should be easily readable on systems other than the minicomputers bought but should be efficient on the minicomputers where most of the use will be. Weighing these and other considerations we decided first that each data tape should be totally self-contained. A description of the tape and a program to read any special formats on the tape should be included on the beginning of the tape in "card image" format. Secondly we decided to store the data in a free format where each set of data was preceded by descriptors of its type and length so that the format is very general. This approach has the added advantage that it is the same way the data are stored on-line in the database, which is described later, so that the tapes can be used efficiently as an off-line extension of the database.

Standardization of analysis programs is a much more different problem. The best chance for standardization is in the routine analysis. Yet even in a program as basic as a least-squares, iterative inversion of travel-time data to determine earthquake locations, there are a variety of valid approaches, some of which work best for some networks and some of which work best for other networks. Given these observations and a consideration of the changing nature of this research environment, we concluded that adequate standardization of specific analysis programs was not feasible, if not impossible.

A further analysis of the problem, however, suggested a viable approach. An earthquake location program, for example, actually consists of about a dozen steps, each of which can be isolated into a subroutine. The variety of approaches to locating earthquakes resolves into a variety of approaches to writing a few of these subroutines. Thus if individual subroutines or modules are written as general "software tools" that can be easily linked, a given scientist can combine the tools of interest with one or two custom built tools and create rapidly a new program to meet the specific needs. Thus the approach we adopted is to build a large library of general purpose tools and to create an interactive environment in which these tools can be readily combined, new tools can be readily fashioned, and documentation of the tools can be readily written and distributed.

While most programmers, in the Earthquake Program at least, are familiar with tools, for example, in a Fortran subroutine library, they are not accustomed to writing their own programs in a similar self-contained manner and are particularly not accustomed to trading these modules. Despite any logic to the alternatives, the momentum is immense to continue programming as we always have. Conversion to the tools approach only seems possible when individuals discover for themselves that they can get more work done faster. This means that significant work must be done to

develop the tool oriented computer environment to a level of high productivity before most people will become seriously interested.

Hardware

Two types of minicomputers were bought for each site: a Digital Equipment Corporation PDP 11/34 for data collection and a PDP 11/70 for data analysis. The 11/34 has 2 TE16 tape drives, 2 RL01 disks, 96 Kbytes of memory, a floating point processor, console and an LPA-11 microprocessor controlled analog-to-digital converter. The hardware configuration of the 11/70 is 2 TE16 tape drives, an RP05 disk, 256 Kbytes of memory, a floating point processor, DH-11 serial ports, console, tektronix 4014 display with DR11-B parallel interface and a Versatec 1200A printer/plotter. At the center of the largest network (containing 512 seismic stations), the 11/34 is being set up in conjunction with an 11/44 and many custom built microprocessors to do the data collection and on-line processing. Most 11/70 systems have been or will be expanded to several disks, more memory, and a variety of peripherals. In one case DMC-11s will be used to link two 11/70's many hundred miles apart at 9600 baud.

Groups interested in using the analysis software being developed have the option of buying an 11/44 instead of an 11/70. Use of the 11/34 for running the analysis software is not recommended at this point.

Operating System

Considerable effort was spent over a three month period researching and deciding on an appropriate operating system. The hardware vendor supplied IAS and RSX systems had the obvious advantages of vendor support, strong real-time capabilities, and good hardware diagnostics. A major drawback was the difficulty within these systems of developing a tool oriented environment where tools could be "dynamically" linked to a command language. Interprocess communication was difficult to use. The UNIX Timesharing System was chosen instead, because it was designed using the tools approach. Inter-process communication between many processes was easier to use. This system performs better than IAS or RSX with a large number of users in Timesharing mode. It utilizes the hardware capability for separate instruction and data address space thus doubling the address space available within any process. The UNIX Timesharing System is written in a high level language "C" which makes it easy to understand, support, and modify to fit specific needs. The need for some realtime capability was easily implemented. This operating system provides a particularly favorable environment for programmers and for utilizing "tools." Some very powerful tools for parsing, compiling, text processing, typesetting, etc. are provided with the system. The UNIX Operating System is the most portable operating system available. It has been brought up on a variety of minicomputers and mainframes. The principal drawbacks with UNIX from our point of view are poor hardware diagnostic messages and a relatively new Fortran compiler. The lack of vendor support has turned out to be much less of a problem than anticipated because of the profound clarity and simplicity of the system. Initially the users were evenly split on the decision to go with UNIX. The users are now overwhelmingly satisfied with the decision. RSX11-M is being used on the 11/34 temporarily until an appropriate driver is written for the LPA11.

Programming Languages

Most scientific programmers, at least in the earthquake program, know and love Fortran and Fortran is the most portable computer language available. For these rea-

sons we started with an overwhelming bias of trying to program everything in Fortran or Ratfor, which improves some of the deficiencies of Fortran. After several years of development in Fortran on CDC 7600 and Honeywell Multics machines and only one year on the PDP 11/70 under the UNIX Operating System, we have concluded that we should have switched almost immediately to writing new code in the "C" Language. This is particularly true for major tools such as the interactive command language, plotting packages, and database system. Many programs written in Fortran and then totally rewritten in C turn out to run one to two orders of magnitude faster. They compile into much less code, they are much easier to write, far easier to understand, and significantly easier to debug and maintain. C makes it easy to write table-driven programs. C essentially documents itself because it is terse and clear. The availability of a portable C compiler that has been brought up on many different computer systems allows code written in C to be quite portable. C compilers are also becoming more generally available. While most of the user written tools are likely to remain in Fortran, C ends up being adopted by everyone in our group who seriously tries it.

Interactive Command Language

Herriot has put considerable thought into the development of a general purpose interactive language. Several versions have been written over a ten year period and a new and greatly improved version is being written for the 11/70 and UNIX. Geolab, as it is called, is an interactive "geophysical laboratory" giving the user an arena in which to shape and test out ideas quickly and conveniently. Whereas the Shell, which is the UNIX command language, provides an excellent environment for development of tools, Geolab delivers those tools within a context dedicated to scientific research. The Shell facilitates manipulation of files, text editors are for textual information, and Geolab is designed to serve scientific data applications. While Geolab grew out of applications in the geosciences, it is equally applicable to all areas of research and most types of computing.

The needs for data analysis in scientific research are extremely varied and demanding. Not only are hundreds of mathematical, database, and plotting operators required even to get started, but it must be possible to add new tools expeditiously. And yet this same environment must be easy to use even for the uninitiated--its users want to focus on their particular applications, not on the subtleties of an interpretive computer language.

A typical simply-stated task (but often tedious to accomplish) is to plot some data from a file. It should not be necessary to write a special purpose program to do this. Even plotting the cosine of all the points in a data file should be easily expressed. By way of example, here is how this would be done in Geolab:

open "mydata"
ploton tektronix
plot 1 (cos getnums)

Geolab is based around a stack that can contain not only numbers but strings, arrays, string arrays, structures, etc. Array arithmetic is direct and natural. The possibility of having missing data points in time-series data, which is a typical problem in certain types of research, is treated automatically in most of the arithmetic and plotting operations. Arithmetic may be done in postfix or infix notation with assignment from right to left or left to right. Take an example where we want to define a new tool or operator called "doit", i.e. a new word in our completely expandable vocabulary. We want to be able to say:

doit data number

where data is an array of time-series

data from each element of which we wish to subtract the value of number, then we will bandpass filter the result, take the fourier transform and plot it on a new page. In Geolab, we would simply type:

op doit (page plot 1 (~ - ~ bandpass filterorder locut hicut fft))

Operators may be defined in terms of other operators so that new vocabularies may be developed easily to allow the user to think in terms of logical units of work rather than functional units of some computer language. Yet the user may also readily dissect any operator to look at each of its parts at whatever level of detail is desired. Any operator may be written in the Geolab language or in any other computer language available on the computer. Geolab language contains all of the flow of control and test constructs found in most popular computer languages. New constructs can be defined easily. Routines can be readily defined for analysis of large amounts of data.

The wide variety of "software tools" in this system are designed to be readily and easily used and combined from Geolab or from a compiled language such as Fortran or C.

Plotting

A high-level, device-independent, general graphics system has been developed by Ward that interfaces closely with Geolab or any compiled language and runs on minicomputers. Geoplot, as it is called, provides device independence, but automatically uses, when appropriate, any special hardware features of a device such as hardware lettering, line dashing, etc. The user has total "hands-on" control of most features of the system from moving or drawing a line to a point, to setting the size, aspect ratio, spacing, and angle of lettering, to setting the width, pattern, intensity, or color of any line, to setting the lengths of the tic marks and other properties of automatic gridding. Transformations are centralized and over 30 are provided including 20 map transformations. While the experienced user can manipulate a wide variety of options, the system is easy for the beginner to use because default values are set up for all options.

Geoplot runs as a separate process within UNIX. Data are sent to it by pipes so that even though it consists of a large amount of code, it requires only 3000 bytes of the users address space. Geoplot may also be treated as a device filter from the UNIX Shell.

Relational Database

A fundamental need of a research scientist is to have easy access to large amounts of data. The heart of this tools approach is a database system that we call Geobase. This system gives the user ready access to any type of data by name rather than by column number in some card image. The relational database can be considered logically to contain a large number of two-dimensional tables or relations where any item in the relation may be a number, an array, a time series, text, etc. Each table can be readily accessed and information from different tables can be collated. Many items are indexed so that searches of these tables can be extremely fast. Thus data or groups of data can be requested in a natural form and delivered without the user having to be concerned at all with how the data are physically stored.

Geobase is now in its third major rewrite. This new version, written in C, requires less than 16 Kbytes of address space for text to provide all of the storage, retrieval, and indexing functions. It utilizes many buffers and caches to minimize disk traffic. We are developing a query language within Geolab. With time Geolab and Geobase will most likely grow into one unified system.

UNIX Extensions

A number of extensions to the UNIX Operating system have been developed primarily by Jolitz. Version 7 UNIX, as distributed by Bell Laboratories, will work only on machines with separate instruction and data (i&d) spaces such as the PDP 11/45 and 11/70. The hints and code provided in the directory **/usr/sys/40** give enough information to make a functional system. The principal problem is the bootstrap, which is harder than it might seem, since the **boot** program on version 7 itself runs separate i/d, and must be run this way to load the separate i/d versions of the system. We modified significantly the machine language startup of the system, and the intermediate bootstrap so that both could run independent of the type of machine. The bootstrap and the system is universal, working on any mapped PDP 11 (e.g. 11/ 23, 34, 40, 44, 45, ... 70). The modifications to the system code are conditional, and makefiles have been made that know about the two different flavors of kernal that are possible. These changes have been running on a PDP 11/34 since July 1979.

One of the most troublesome aspects of the PDP 11 architecture is that its addressing space is too small. Even with separate i&d spaces, one is still limited to 65K bytes of instruction. For this reason many programs have to be run as clusters of processes, tied together by many pipes. About a year ago C. Haley and W. Joy worked out the theory for an unusual overlaying method to increase the size of programs on PDP11's. We have implemented this scheme after substantial modification of their work, and are able to run 200 Kbyte text programs on standard PDP11's without much loss of efficiency. The overlaying scheme is simple because it avoids using any overlay description language, quick in that overlays can be loaded in 100 microseconds, and invisible to the program. The system as it now exists does not swap unused overlays out of memory, but future revisions will, allowing programs larger than physical memory to run on the system. Currently we have versions of the Berkeley editor **ex** (version 2.0) and of **f77** which run on the PDP 11/34 UNIX system. We are working on a version of **adb** to allow interactive debugging overlayed processes.

A relativly new and untested feature to our system is the addition of a limited real time capability based on work done several years ago by David James at Lincoln Laboratories. These modifications allow one real time process to run while time sharing is active.

Two new drivers have been added to the system. An RM02/3 driver is provided which has been tested on various systems, and even works with non-DEC RM02 lookalikes. A driver has been made for a fast tektronix interface (described elsewhere by Ellis) that allows for raw dma transfers from the users' buffer to a given Tektronix at 830,000 baud (e.g. UNIBUS rate limited). This driver has been designed to appear as a normal tty port with the exception of **ioctl** modes which can cause output to be routed via a DR11-B parallel dma controller.

As mentioned above, the stand-a-lone system bootstrap was modified to become universal on PDP 11's. Additional capibility was added to allow other discs to be used (RL01, RK06/7, RM02/3), and appropriate initial block bootstraps were written also.

The other most significant changes that have been made to the distributed version 7 software are:

1. A number of intrinsic functions were added to **f77** for moving bits.

2. David Wasley's corrected Fortran i/o library (**libI77**) was implemented on the PDP11.
3. **init** has been rewritten to support a more useful /etc/ttys file format that includes terminal type and location.
4. **su** and **newgrp** have been extended to know about different kinds of shells using the SHELL environment variable.
5. Unconverted version 6 programs included in the distribution (**cu,pstat,...**) have been converted.
6. A **man** command has been written that is table driven to allow easier tailoring to local needs.
7. Two conversions have been added to the **dd** command for reading and writing ASCII files. **rtrim** and **rfill** add a new line and trim right blanks or delete the newline and add right blanks respectively.
8. Several separate i&d programs have been re-engineered to run non-separate on 11/40 style machines.

Many of these items are included on the conference distribution tape.

Manuals

A key to the tools approach is to have good documentation readily available. The UNIX Timesharing System provides excellent tools for this purpose and all documentaton is available on-line as well as in typeset hardcopy form. Each tool is individually documented and a number of papers are written about groups of tools or large tools such as Geolab. The manual system for UNIX and tools is integrated into 8 volumes containing over 2000 pages.

Availability

All of the software being developed will be available at the cost of copying. Geolab, Geoplot, and Geobase are written and functioning along with a large and ever expanding library of tools. While the software has been distributed in various forms to the sites that we are responsible for supporting, it is not yet refined enough to be of general use. Substantial amounts of rewriting, changing and tuning are underway and we do not plan to distribute the system widely until at least late in 1980. The software is designed for DEC PDP 11 machines from 11/44's to 11/70's and requires the UNIX Timesharing System.

A VAX version is planned in 1981.