CRAY USER GROUP

INCORPORATED

PROCEEDINGS
Spring 1986

CRAY USER GROUP
INCORPORATED

PROCEEDINGS

SEVENTEENTH SEMI-ANNUAL CRAY USER GROUP MEETING

May 5–9, 1986
Seattle, Washington

Host: Boeing Computer Services Company

KAREN WINGET, EDITOR

CONTENTS

Monday, May 5	
10.00 -12.30	Operating Systems SIC Closed Meeting
10.00 -12.30	Software Tools SIC Closed Meeting
10.00 -12.30	Operations SIC Closed Meeting
10.00 -12.30	Communications SIC Closed Meeting
11.00 -12.30	Graphics and Databases SIC Closed Meeting
	[Observers may attend SIC meetings with the prior permission of the SIC chair]
1.30 -3.00	CTSS SIC Closed Meeting
1.30 -3.00	Performance SIC Closed Meeting
2.00 -5.00	Aerospace Birds-of-a-Feather
3.00 -5.00	User Requirements Committee
5.00 -6.00	Advisory Council
7.00	Cray Corporate Reception

Tuesday, May 6	
8.30	Welcome A.M. Erisman (Boeing)
8.45	Keynote D. Ritchie (Bell)
9.15	CUG Report H. Kulsrud (IDA)
9.30	CRI Corporate Report M. Gumucio (CRI)
10.00	Break
10.30	CRI Software Report M. Loftus (CRI)
11.00	UNICOS on the X-MP J. Miller (CRI)
11.30	Cray-2 Development Status D. Judd (CRI)
12.00	Lunch
1.30	Software Tools I C on the Cray M. Zosel (LLNL)
	Performance I IO M. Lewis (Chevron)
3.00	Break
3.30	Software Tools II Libraries C. Lazou (ULCC)
	Operations I G. Jensen (NCAR)
5.00	Reception for New Members
	UNICOS Demonstration (CRI)
8.00	Tour of Boeing Data Center

Wednesday, May 7	
8.30	Operating Systems I UNICOS R. Benoit (EC)
	Operations II G. Jensen (NCAR)
10.00	Break
10.30	COS-UNICOS migration R. Lagerstrom (CRI)
11.00	Aerospace & Supercomputing P. Muzio (Grumman)
11.30	The Supercomputer as an expmtl. apparatus B. Domenico (NCAR)
12.00	Lunch
1.30	Communications I UNICOS Station Software D. Smith (ARCO)
	Graphics H. Kulsrud (IDA)
3.00	Break
3.30	Performance II Performance Evaluation A. Cowley (NCAR)
	Communications II Superlink D. Smith (ARCO)
5.00	Program Committee Software Tools SIC Meeting
7.00	Conference Dinner

Thursday, May 8	
8.30	Communications III User Presentations D. Smith (ARCO)
	Software tools III Multitasking E. Williams (LANL)
	Operations III G. Jensen (NCAR)
10.00	Break
10.30	Cray-2 s/w enhancements T. Hoel (CRI)
11.00	Short Papers C. Kimble (Boeing)
12.00	Lunch
1.30	Operating Systems II COS R. Benoit (EC)
	CTSS J. Melendez (LANL)
3.00	Break
3.30	Advisory Council CFT77 Status Karen Spackman (CRI)
4.00	User Requirements Committee
5.00	UNICOS demonstration (CRI)
5.30	OSSIC Closed Meeting

Friday, May 9	
9.00	New Hardware Performance C. Diem (CRI)
9.30	Implementing UNICOS J. Barton (NASA Ames)
10.00	Break
10.30	Reports from Special Interest Committees and User Requirements Committee
11.15	29 Million digits of D. Bailey (NASA Ames)
11.45	Next Conference P. Herchenbach (DFVLR)
11.55	Closing Remarks
12.00	Lunch (not sponsored)
2.00	Tour of 747 factory

PRESENTATIONS

MARKETING OVERVIEW

MARCELO A. GUMUCIO

CRAY RESEARCH, INC.

REVENUE

1981......101.6

1982......141.1

1983......169.7

1984......228.8

1985......380.2

NET EARNINGS

1981.......18.2

1982.......19.0

1983.......26.1

1984.......45.4

1985.......75.6

R & D EXPENDITURES

1981.......17.0

1982.......29.5

1983.......25.5

1984.......37.5

1985.......49.2

WORKING CAPITAL

1981.......61.9

1982.......98.6

1983......129.4

1984......115.8

1985......137.9

CAPITAL EXPENDITURES

1981........9.3

1982.......19.3

1983.......14.9

1984.......25.7

1985.......49.7

TOTAL EMPLOYMENT

1981......1,079

1982......1,352

1983......1,551

1984......2,203

1985......3,187

CUSTOMER BASE

	Total Customers	Total Installs
1981	25	35
1982	32	50
1983	44	65
1984	60	88
1985	79	115

1985 INSTALLED BASE BY INDUSTRY

Petroleum	16%
Service Bureau	6%
Aerospace	13%
Weather	4%
Universities	6%
Automotive	3%
Electron	5%
Government	38%
Other	9%

CRAY SOFTWARE STATUS

Margaret A. Loftus

Cray Research, Inc.

Since the last User Meeting, we have distributed 22 software releases. This includes 12 major releases and 10 minor releases. Eight releases were distributed in March and April and includes some of our major products.

The following software has been released since the last Users Meeting.

3.02 VAX/VMS Station	October 1985
3.0 VM Station	November 1985
1.15 NOS Station	December 1985
Superlink/ISP 1.0	December 1985
UNICOS 1.0	March 1986
CAL 2.0	March 1986
PASCAL 3.0	April 1986
1.0 C Compiler	April 1986
1.16 NOS Station	April 1986
2.0 UNIX Station	April 1986
COS 1.15	April 1986
CFT77 1.0	April 1986
PREMULT prereleased	January 1986

The following software is running at Cray sites:

	No. of Machines
COS 1.15	8
CFT 1.15	8 (prerelease)
COS 1.14	67
CFT 1.14	72
COS 1.13	22
COS 1.12	6
CFT 1.11	24
CFT 1.10	1
UNICOS	9
CFT2	6

COS and CFT 1.14 are being on most machines.

UNICOS use is just beginning but is increasing rapidly. Last year we estimated 1986 UNICOS users at all 1986 CRAY-2 customers and 3 X-MP customers. The number of UNICOS X-MP customers is already three and we have currently a total of 14 strong prospects for 1986.

The following are the major software projects in progress today.

- 1.15 CFT planned for release in May.
- 1.16 COS/CFT will be released late 1986/early 1987. Major features include:
  - Guest operating system enhancements
  - Internal Station - COS to UNIX Station
  - Ability to partititon the SSD and online tapes.
  - Microtasking officially released
  - SSD backdoor
  - SSD and buffer memory preemption or rolling
  - New NSC driver - improved error recovery
  - Concatennated datasets
  - Also a significant emphasis on reduction of SPRs.
- CFT77 for the CRAY-2 will be released in July. Beta testing at a CRAY-2 site has begun.
- 1.15 NOS/BE Station to be released July with dual state support.
- Superlink/MVS 2.0 will be released 4Q86. We have also begun a Superlink effort under UNICOS.
- 2.01 MVS Station to be released in May with interactive support. We have just finished beta testing.
- 2.0 Apollo Station to be released May with operator commands and control.
- UNICOS 2.0 will be released 3Q86 with job recovery, support for SCP protocol.

A major emphasis of our software is transportability software. 1986 is the year we begin seeing results from that effort with CFT77 and UNICOS.

CFT77, a new FORTRAN compiler has been initially released. Characteristics of CFT77 are:

- Written in higher level language - PASCAL.

- Base for improved performances
  - incorporates new techniques for vectorization/optimization
  - improved scalar performances. Base for automatic multitasking.
  - Compatible with the existing compiler - CFT.

Our transportable operating system UNICOS 1.0 was released last month. Capabilities are:

- Supports both the CRAY-2 and X-MP.
- Written in C
- High performance - equivalent to COS for I/O and CPU overhead for the X-MP.
- Extensions to UNIX - I/O and batch
- TCP/IP support

New releases of UNICOS will occur about every 6 months with mini releases every 2 to 3 months.

In early 1988 we expect UNICOS to have the functionality of COS. UNICOS and TCP/IP both require licenses.

The introduction of UNICOS requires a strong migration effort. Migration provides a bridge from COS to UNICOS for those customers interested in migrating.

Our goal is to have a single product FORTRAN compiler, CFT77, and a single operating system, UNICOS, that runs on all the hardware that we are offering.

Anothers major software emphasis is connectivity. Connectivity is important because our software must allow CRAY systems to fit easily into diverse environments.

Stations will continue to be an important means of connecting to CRAYs. Stations will provide connections to both COS and UNICOS. Our station products will continue to be enhanced as needed.

Fitting into networks is a major effort. There are two ways of fitting - as equal partners and through a frontend gateway. Stations and the Superlink effort will provide frontend gateways to a variety of proprietary networks such as DECnet and SNA.

For true network environments, TCP/IP under UNICOS is available for both the CRAY-2 and X-MP.

A third major software emphasis is improved quality. A number of changes are being made to address quality:

- More field testing prior to formal release. More frequent bugfixes (every 3 months).

- There is a concentrated effort on improving on-line tape reliability by more field testing, improved testing in Mendota Heights and improved access to a variety of configurations.

- Additional hardware in Mendota Heights.

- Controlled introduction of major new products.

- Faster response to major SPRs.

A user survey has just been completed surveying 45 customers. We plan to respond to the concerns highlighted in the survey and plan to repeat the survey in 1987 to see if you think we have improved.

UNICOS on the CRAY X-MP

Jim Miller

Manager, X-MP UNICOS Development, Cray Research, Inc., Mendota Heights, Minnesota.

Introduction

> "*... one of the most important aspects of any computing tool is its influence on the thinking habits of those who try to use it ...*" *E.W. Dijkstra.*

UNICOS is Cray Research's operating system for the Cray-2 and is available for the Cray X-MP. UNICOS is based on AT&T UNIX† System 5 but has some software from 4.2bsd and many changes and additions from Cray Research to allow it to run successfully in a supercomputer. The first release and subsequent releases are common to the Cray-2 and X-MP. There are differences between the software that run on the machines, but the bulk of the code is common and offers a common user interface to applications programs.

The first port to a Cray machine was from source targeted for a Vax. The software now runs on Cray X-MPs with 2, 4, 8, and 16 million words of memory and 1, 2, or 4 processors; Cray 1/Ms and Cray 1/Ss as long as an I/O Subsystem is attached; and also Cray-2s.

Release 1.0

UNICOS was released in March and is available to Cray customers with the necessary licenses. Some Cray-2 and X-MP customers are using UNICOS at this time. At this writing there are nine Cray machines which are running UNICOS under the Guest Operating System or in native mode. The next release of UNICOS will be in the third quarter of 1986.

† UNIX is a trademark of AT&T Bell Laboratories.

The major features that have been added to the System 5 standards are:

- TCP/IP protocol support. Software to communicate with front ends with similar support.
- Network Queuing System. A subsystem that allows batch processing under UNICOS.
- Performance Enhancements. Changes to allow better I/O throughput and improved operating system performance.
- Multiprocessing support.

X-MP UNICOS Goals and Requirements

UNICOS was modified to satisfy several sometimes contradictory goals. The main goals pursued are the following:

- Stability.
- Performance.
- Compatibility with Cray-2.
- Compatibility with AT&T System 5.
- Common Products with COS for migration purposes.

Stability

The structure of UNICOS is well suited to avoid system interruptions. The small kernel is easy to maintain. Implementation in the higher level language, C, gives us inherent features that grant better stability; modular structure gives ease of modification and localization of functions.

Most errors are encountered as a result of user action and therefore the typical reaction at time of unrecovered errors is to abort the user and allow the rest of the system to continue. The UNICOS command set includes a large share of what is often considered operating system in other systems. Those commands are essentially unchanged since the port from the Vax, mostly as a result of the effort to maintain a compatible user interface with System 5. The commands are very mature software and give us a predictably stable environment.

Performance

Compute bound programs are able to perform well because the hardware features continue to be used by the COS compilers and products that have been moved to the UNICOS system. Code produced by these compilers performs virtually identically to the same code running under COS. The performance of I/O devices is one of the most important concerns of an operating system. For UNICOS most of the devices in use are attached to the IOS which runs mature software originally developed for COS. For DD29s, DD39s, and DD49s the performance delivered by UNICOS equals that of COS. Software in support of SSDs resides in the main frame, but when compared to I/O done in COS to SSDs, UNICOS performance is measurably better. This is due to shorter paths in UNICOS to get to the device from the user request. UNICOS is especially effective in smaller size requests. Buffer memory resident data sets are comparable to COS speeds because they also use the IOS software to drive the device. In addition, hyperchannel and basic on line tape software support exists in the first release via the IOS.

Cray-2 Compatibility

Cray-2 compatibility is an important goal. A single operating system across all Cray main frames allows significant economies of effort in producing and supporting systems. The hardware differences are significant so that several software differences have been necessary.

The Cray X-MP has an IOS attached, memory to 16 million words, SSDs, a separate base address and limit for code and data, and cluster registers.

The Cray-2 has a front end processor, immense memory (256 million words), and a single base address and last address register, in addition to a different instruction set.

UNICOS for the X-MP will be compatible with UNICOS for the the Cray-2. The goal is to have the same interface for user calls, system calls, library calls, and languages. Exceptions will occur due to architectural differences or to performance needs which are dependent on architectural differences.

AT&T System 5 Compatible

Compatibility with System 5 is another of the important goals of the project. The user interfaces and general environmental structures of UNIX were preserved in the port. UNICOS, like UNIX, has a small kernel that is well insulated from user activity. The bulk of work is done at user level.

The System V Interface definition (SVID) manual as written by AT&T is being used in the project. It defines the minimum set of functionality included in a System 5 UNIX. We plan to meet the criteria set out by AT&T.

The UNICOS system is based on a UNIX that ran on a much smaller machine and thus many changes were necessary in the system to allow much better performance. The following paragraphs describe the major changes to accomplish this.

Asynchronous I/O was implemented in UNICOS to allow a user to overlap computational work with I/O. X-MP UNICOS supports reada and writea commands as well as a list directed call, listio, which allows multiple distinct I/O requests to be included in a single request.

I/O block size by file system is supported. System 5 for the Vax uses a block size of 1024 bytes which is too small for good I/O performance on a Cray. For X-MP UNICOS the system allows each file system to specify a larger block size which is saved in the file system descriptor and used in all I/O requests on that file system. In practice that block size is set to at least a sector (4096 bytes) and for large files needing good streaming rates it's set at a DD29 or DD49 disk cylinder (18 or 42 sectors)

File system changes have been made to enhance the performance profile of the system. The allocation mechanisms have been changed to reduce file fragmentation which, in turn, increases the system's ability to stream data from files. The maximum size of a file has been greatly increased by software which allows device overflow through the use of the concept of a logical disk.

Raw I/O is extended to normal files by the system. Raw I/O allows faster I/O by movement of data directly to the user buffer rather than via system buffers. Transfers are done in whole sectors. Raw I/O is selected at file open time which causes all normal read and write calls to be raw I/O requests.

Multitasking and microtasking are supported in a COS like implementation. Both functions are effective means to allow a single job to use the processing power of more than one CPU concurrently. A preprocessor (PREMULT) for CFT is available by request and on a prerelease basis to support microtasking.

Other Changes to UNICOS

Disk flawing support has been added to let the system allow for imperfections in the disk media.

Batch capabilities which are based on Network Queuing System (NQS) are included in the release. Capabilities to submit, manage, status, and delete jobs from queues are available via UNICOS commands.

Accounting changes have been implemented to be more suitable for supercomputer usage.

Terminal support in UNICOS is resident in the front end and relies on TCP/IP to get terminal input to the UNICOS system.

Common products with COS

CFT 1.15 and Fortran libraries. The 1.15 compiler is a prerelease which will run under UNICOS.

C 1.0 compiler and libraries (libc) are based on C language V.2 from AT&T. Most differences arise from hardware dependencies.

Pascal 3.0 and libraries. The compiler has been enhanced over 2.0 and runs under UNICOS.

CAL 2.0 is a new version of the Cray assembler written in Pascal to allow it to support all Cray machines. The command name for CAL Version 2 under UNICOS is *as*.

SEGLDR 2.1 is the COS SEGLDR modified to run under UNICOS.

UPDATE 2.1. UPDATE under UNICOS performs the same functions as under COS and is used as source control utility for some of the UNICOS products.

APML, ADSTAPE, BIND. These products are necessary to be able to build the IOS software under UNICOS.

DEBUG 2.1 is the COS DEBUG program modified for UNICOS.

IOS software. The full set of IOS software is able to run with UNICOS as well as COS.

Commands and System Calls

The first release of UNICOS contains over 250 commands including System 5 standard commands that are supported on most UNIX main frames, and a set of Cray specific commands that have been produced to support specific features that have been ported to UNICOS or created to deal with specific supercomputer related issues.

The standard commands include such commands as *awk, crypt, grep, and man*; a set of commands that is normally found in every Unix implementation. The list includes COS products: *apml, cft, debug, pascal, update,* along with several others. These are familiar to COS users and perform as they do under COS. The set of commands created by Cray and ported to Cray machines for specific uses include TCP/IP commands (*rcp, telnet*) and NQS commands (*qsub, qstat, qdel*).

There are over 80 UNICOS system calls that provide services to running programs. They include standard calls and some that have been added to get specific functionality. Some examples of system calls provided are *open, close, reada, writea, listio, fork, and wait.*

COS TO UNICOS MIGRATION

Richard N. Lagerstrom

Cray Research, Inc.
Mendota Heights, MN

OVERVIEW

This paper will describe what Cray Research is doing to help current COS users migrate to UNICOS. This is not intended to be a technical overview but a general description of what is or will be available from CRI. As of the date of this paper little actual experience with migration of users exists. We intend to further develop migration aids as needed and suggestions are welcome.

The subjects covered below are 1) What products are the same between COS and UNICOS; 2) Migration Aids available; 3) Training and 4) Migration Coordination.

Migration from COS assumes current use of Cray 1 or Cray XM–P machines. The migration target includes the Cray 2.

WHAT IS THE SAME?

The major products listed below are common between COS and UNICOS. A common product is source compatible between operating systems and provides the maximum degree of migration ease between systems.

Compilers

The CFT, Pascal and C compilers are common between COS and UNICOS.

Libraries

The Fortran libraries are the same except where operating system differences make that impossible. The names of the libraries are different in the two systems as shown in the following table

$SCILIB is named	**libsci** on UNICOS
$ARLIB is named	**libm** on UNICOS
FORTRAN–77 I/O is named	**libio** on UNICOS
$UTLIB is named	**libu** on UNICOS

Loader

SEGLDR 2.1 is the common loader on COS and UNICOS.

UPDATE

The source maintenance utility UPDATE is common between COS and UNICOS. A utility named **plcopy** is used on UNICOS to convert a source file from COS format.

MIGRATION AIDS

A number of migration aids are being developed to make migration from COS to UNICOS easier. This section of the paper will describe the currently identified migration aids in functional terms. More detail may be requested from your field representative.

Guest Operating System

GOS is the Guest Operating System which is a feature of COS released in 1.15. This feature allows UNICOS and COS to run together in the same machine given that sufficient resources are available. The general requirement is that UNICOS requires one million words of memory and at least one dedicated disk, one CPU and an IOS. This means that the version of the Guest Operating System available with COS 1.15 will not run on single CPU machines. The CPU assigned to UNICOS is returned to COS when UNICOS is stopped.

To increase the utility of GOS a number of characteristics have been changed and will be available with COS 1.16 and 1.17. The improved features allow use of GOS in single CPU machines as well as providing demand scheduling for multiple CPUs under installation control. Also included is a COS to UNICOS connection for direct communication between the systems using SCP and the ability to partition disks and the SSD between systems. Because some of these features depend on a specific version of UNICOS, please contact your field representative for release details.

The COS 1.15 version of GOS has been designed to isolate the two systems as well as feasible in the COS environment. Each system has its own region of memory protected by the system's BA and LA registers. The only code able to access both portions of the machine is COS EXEC. This is necessary since interrupt processing for UNICOS is done in part by EXEC.

The fully protected mode of GOS mentioned above is the only mode available in COS 1.15. This mode requires memory allocation by STARTUP so the memory may not be restored to COS unless STARTUP is run. Some installations have indicated that it would be better to dynamically allocate memory to UNICOS when it is loaded thereby leaving the memory for the use of COS when UNICOS is not in use. COS 1.16 will have this feature so memory is allocated to UNICOS when it is loaded and returned to COS when UNICOS is stopped. Both modes are available in 1.16 but only one may be in use at a time. The dynamic mode is not protected from COS but this does not appear to be a serious problem.

Control of UNICOS is through a number of simple commands running on COS. The operator or a privileged user may load, dump or stop UNICOS using the COS commands. The COS operator also has access to UNICOS through the IOS terminals.

JCL Conversion

A number of utilities are available on both COS and UNICOS to help convert COS JCL to UNICOS scripts. The output of JCL conversion is suitable for direct execution or it may be submitted through NQS (the Network Queuing System or batch execution facility).

The JCL conversion process takes a file of COS JCL as input and produces a UNICOS script file as output. Procedure libraries are also supported with a preprocessor which converts a COS PROC library into a form useful to the JCL converter. The translation of a JCL line uses a set of directives referenced by the COS verb. The directives are stored in a file and this file may be changed by users to more closely reflect the specific environment. The set of directives as released support standard COS statements. These may be extended as needed to support other commands.

Any statements not converted are marked in the output and will be visible to the user. If a converted script with unconverted commands is executed, an error will be reported when an unconverted line is encountered. Error checking is also available on a command basis. Either cause of errors during execution will result in a message telling on which line the error occurred if in the main script or the line and PROC name if the error occurred in a procedure.

PDSDUMP Conversion

Two utilities are available on UNICOS to convert COS PDSDUMP files into UNICOS **ar** format files. The utility **ptoa** converts the file format itself while **ctou** is used to convert blocked files, multifile datasets and files with blank compression.

Documentation

A large amount of documentation has been written to support UNICOS both for user, operator and site administration purposes. Common products such as compilers and the loader have new documentation which describes their use on both systems and notes differences between systems.

For migration a series of case study technical notes is being developed as information becomes available. The purpose of this is to capture migration experiences of the user community so information developed in one place can be of benefit to all users. We rely on input from field personnel for the content of these technical notes. We encourage forwarding experiences (both good and bad, successful and unsuccessful) to the field representatives for inclusion in technical notes. Exact technical details are not always required – often ideas and suggestions in general terms are just as valuable. This means that it is not necessary to include sensitive information to contribute to this collection of experience.

Work is also in progress on conversion guides. The purpose of this type of documentation is to show how to go about migration of such things as CFT programs and JCL.

TRAINING

Since UNICOS is derived from UNIX† System V courses available for that system are applicable to UNICOS. These courses are often available from local sources and are generally useful for UNICOS users.

Training courses have also been developed for UNICOS and are integrated into the standard training program. These are specifically targeted for UNICOS users and emphasize the characteristics of the system specifically designed to exploit the performance potential of Cray computers. Training on the internals of UNICOS is also a part of this series of courses and is important to anyone having a need to know the implementation details of UNICOS.

Migration training courses are also available.

COORDINATION

There is much going on within the company in the area of migration so to keep track of what is happening a Migration Team has been established to oversee this work. The team includes members from the various technical groups within Cray Research and at least one member from each region and country. The region and country members are the easiest path from the user community to the Migration Team and we encourage them to bring the needs of their user community to our attention.

† UNIX is a trademark of AT&T Bell Laboratories

The Supercomputer as an Experimental Apparatus: Some Thoughts on Scientific Supercomputer Centers

Ben Domenico*

Science Horizons, Inc.
7100 Encinitas Boulevard
Encinitas, CA 92024

OVERVIEW

The first part of this paper is an attempt to examine the role of a scientific supercomputer center from a scientist's perspective. Initially the guiding principles and overall goals are laid out, followed by a discussion of the tasks performed by scientific research groups. The discussion of these tasks focuses on areas where computer automation might increase scientific productivity. Based on the perceived needs of the scientific user, the closing sections of the paper propose a distribution of computing functions among a network which includes the scientist's workstation, divisional or departmental minicomputers, and supercomputer centers.

GUIDING PRINCIPLE

For the NCAR Scientific Computing Division (SCD) as well as other groups providing and purveying scientific computing services, the broad guiding principle is to insure that, in the long term, those services will increase scientific productivity. While this may seem obvious, it is not always obvious that concerns such as making efficient use of computing machinery and keeping up with technological developments-while they are certainly important-derive their importance from the primary goal of improving scientific productivity. If there is a trade off between wasting machine cycles and requiring a scientist to spend extra time to rework a program, one had better carefully weigh the alternatives. Likewise, if a long period of retraining is necessary to take advantage of a new technology, it's important to establish that the new approach will have a long-term positive effect on scientific productivity.

OVERALL PERSPECTIVE

A traditional approach to determining how computing services can increase scientific productivity is to examine the available technology first and then try to determine how that technology can be applied to scientific research. Another approach is to examine what it is that a scientist and her support staff do during their workdays. With such a list of tasks, we can then look for computing services that might help the research group get the work done more efficiently.

The approach taken here is to list the major tasks performed by scientists, break down the tasks into components, and determine which aspects of those tasks can be performed more efficiently through some form of computerization. One advantage of this approach is that tasks can be prioritized and resources allocated according to whether computerization would result in significant increases in productivity. Assuming that some form of automation is possible and warranted, another issue is whether the computerization should be done at a central site, within the scientist's division or department, or even on the scientist's own personal computer. In cases where the function or service itself should be distributed, there is the question of whether a central source of research, information, expertise, coordination, standardization, and or purchasing power may be needed.

For Example

As an example, at NCAR, there is general agreement that the production of technical documents lends itself well to automation. Although there seems to be a consensus that the function should be performed on divisional minicomputers or personal computers and workstations, an office automation study group survey revealed a need for standardization on products, a central source of documentation and consultation, as well as cooperative projects involving the communication of documents. This is a good example of an automation function that should be distributed, but which benefits from a central source of information and coordination.

SCIENTIFIC TASKS

The appendix of this paper describes in some detail the tasks that comprise the typical work week of a scientific research group. For most such groups, the week is taken up with many of the following endeavors:

1. Constructing scientific theories.
2. Teaching classes and managing graduate students.
3. Developing and running laboratory and field experiments.
4. Developing and running modelling experiments.
5. Storing and accessing archived data.
6. Analyzing data from experiments, models, and archives.

* Former affiliation: University Corporation for Atmospheric Research.

7. Accessing information, reading literature, reviewing publications and proposals.

8. Preparing presentations, proposals, and publications.

9. Attending meetings and communicating with colleagues and staff.

Task Components

In breaking down these tasks into components, certain characteristics warrant special attention:

- elements of tasks which lend themselves to computerized automation.
- requirements which span several of the tasks. For example, improved data communications facilities might improve productivity in many of the tasks.
- requirements of one task that might restrict options in others. For example, real time response needed for computer control of a satellite experiment might have an impact on use of a computer for time-shared interactive access.
- items which are common needs of many scientists. For example, if many groups are developing the same type of software, it might make sense to centralize or at least coordinated the development of a portable version of the software to avoid duplication of effort.
- requirements that involve software that is so expensive or rare that centralized resources are needed.
- items which consume large percentages of staff time. These are the ones where automation can have a significant effect.

Refer to the appendix for more details.

DISTRIBUTION OF SERVICES

Certain patterns emerge when examining the computer automation associated with these scientific research tasks. Perhaps the most obvious is that many of the tasks that lend themselves well to computerization are done best on a local machine, where the scientist and support staff have more control over the computing environment and have convenient access to the final output.

Local Computing Services

The most obvious case of a local computing task is the production of materials for presentations, proposals and publications. Another task that is often done more effectively on a local computer is data analysis. If the data reside nearby, the scientist can examine them interactively. For example, a scientist looking through graphical representations of data might find an interesting feature in one picture that leads her to want to view the same feature from a different angle, or perhaps to look at the same phenomenon at an earlier time, or a different physical parameter at the same place and time. This ability to decide what you want to look at "on the fly" requires fast, interactive access to the scientific database.

In this category of tasks that can be done well on local computer systems are the following:

- Teaching classes and managing graduate students.
- Analyzing data.
- Preparing presentations, proposals, and publications.

Hybrid Computing Services

A second group of tasks falls into a category that has aspects that are done best locally, but requires some connectivity to other machines or at least shared access to files. For example, it's very handy to do a scientific paper on a personal workstation, but it's also important to be able to send a copy electronically to a colleague for expansion or revision. If both collaborators share a minicomputer, this is usually fairly simple. If they work on separate workstations, it's important to have a some form of communications link that provides convenient file sharing, file transfer or electronic mail.

The tasks in this category are:

- Accessing literature, publications and proposals.
- Communicating with colleagues and staff.
- Storing and accessing archived data.

Centralized or Specialized Computing Services

The remaining tasks, while they don't by their nature imply a shared resource, often cannot be done on the same computing system used for other tasks because they require specialized or very expensive equipment. These include:

- Developing and running laboratory and field experiments.
- Developing and running supercomputer modelling experiments.

Even if a field or laboratory experiment is controlled by a computer, the requirements for the control computer are often such that it may not be suitable for the other tasks a scientific group has to perform. For example, a computer set up to ingest satellite data in real time, may not be usable during those periods when the satellite is sending its bursts of data.

Numerical modelling experiments often require supercomputing resources. While some supercomputer sites do in fact use supercomputers for the majority of their scientific tasks, there are certain tasks that today's supercomputers are simply not very good at. In particular, supercomputers are not very good at the fast context switching required by interactive user input. Rolling out a job that fills all of memory in order to respond to cursor input from a user can present some interesting problems on a supercomputer. The simple requirement of refreshing the screen for a full-screen editing session is formidable enough that most supercomputer centers don't have screen editors running on supercomputers. The result has been some of the first innovative distributed applications programs where the supercomputer edits the file while a personal computer edits a few screenfuls of the file on a PC.

Others would argue that tasks like text processing should be done elsewhere in cases where supercomputing resources are scarce.

RECENT DEVELOPMENTS

There are projects underway that will enable scientists to distribute their computing load in a way that suits their individual needs. Vendors, scientists, and computer center managers should be aware of these developments, so that they all can make use of the evolving technology.

The projects in question are:

- the UNIDATA Project at the University Corporation for Atmospheric Research,
- the USAN (University Satellite Network) experiment in SCD at NCAR,
- Project Cypress run by a consortium of universities and corporations,
- the NSFnet initiative of the National Science Foundation.

The common element in all these projects is to provide a standard mechanism for building a wide area network by connecting local area networks at academic and research centers throughout the nation. Initially all the projects are based on the DARPA/TCP/IP protocols and eventually they are all committed to the emerging ISO Open Systems Interconnect (OSI) protocols. UNIDATA and USAN are working toward relatively high-speed satellite links between remote LANS, whereas NSFnet is starting with land-line connections between supercomputer center LANs. Cypress is aimed at providing a lower speed, lower cost alternative with similar functionality.

EMERGING PICTURE

As these experiments become operational, it will be possible to experiment with a distribution of tasks such as the ones described above. The net result will be an environment in which the scientist's workstations can be viewed as the center of her computing universe.

- Good electronic mail facilities will be easy to build using standard protocols.
- Relatively high-speed access to centralized data sources are possible if the databases are accessible on machines with the standard protocols.
- Access to supercomputers will be straightforward if supercomputer vendors provide an interface using standard communications protocols.
- If the scientist chooses a local computing system that can be tied into a local Ethernet with TCP/IP-based communications, that LAN could be connected to a wide area network which will include computers controlling field and laboratory experiments, major data centers, supercomputer centers and possibly most of the machines used by colleagues at other institutions.

In this picture, the scientist can tailor the working environment, so that papers and presentations are prepared locally, possibly using shared laser printers and color cameras on the LAN. Data from any experimental apparatus can be brought to a file server on the LAN and examined interactively on a local workstation. Machines on the same local network can be used for classes, demonstrations or student experiments. Output from supercomputer models can be compared to satellite observational data, using scientific graphics output on a local workstation.

From this viewpoint, the supercomputer, and the supercomputer center become just another node on the network. The scientist need only learn the operating system on her own machine and that on the supercomputer. There is no need for an intervening frontend system at the supercomputer center to complicate the picture by forcing the scientist to learn yet another computing system.

Operating System Consistency

As more vendors standardize on operating system interfaces and communications protocols, the productity gain for the scientist is even greater. With UNIX* available on most hardware from PCs to graphics workstations, through minicomputers, the availability of UNIX on supercomputers completes the picture and simplifies it greatly in the long term for most scientific users. The advent of real time UNIX machines makes it possible to have the same operating system interface to the computers controlling experiments or receiving experimental data. Since most UNIX machines now come with the DARPA TCP/IP protocols, they fit right in with the wide area networks being developed.

* UNIX is a trademark of Bell Laboratories.

Future Possibilities

In the UNIX world (and to some extent with MSDOS** and VMS*** as well), facilities are available for building distributed applications. One particular application that is gaining a foothold is the Sun Network File System (NFS). This network file system is based on a Remote Procedure Call (RPC) and External Data Representation (XDR). The NFS actually enables a user on one machine to access files on another machine transparently. In other words, you can run an editor on your machine while you're actually editing a file which resides on the disks of another machine.

Furthermore the RPC enables the user to build applications that in essence contain subroutine calls for routines that run on another computer. One can imagine such applications where the graphics portion of the application runs on the local workstation and the numerically demanding routines run on a supercomputer. The XDR and RPC have been demonstrated on both UNIX System V and 4.2 BSD as well as MSDOS and VMS. NFS also runs on all these systems although only in client mode on MSDOS. Whether these facilities ultimately emerge as genuine industry standards remains to be seen, but there is clearly a need for this sort of functionality.

In the long term, the picture is one of a scientist logging into a personal workstation with high-speed network access to supercomputers, data archives, and observational experiments. The access to the supercomputer is transparent in that the applications program runs on the scientist's own machine, but the computationally intensive portions of the problems run on the remote supercomputer. Likewise data archives on other computers can be accessed as if they were part of a database on the scientist's own machine.

Time Frame

Given the current confusion and complication in the area of computer communications systems, some of this may seem a bit far-fetched, but the initial USAN connections have already been made, so testing has begun. The remaining USAN sites and the initial set of NSFnet supercomputer sites probably will online within a year. At that point, it should be possible to evaluate the feasibility and limitations of the system and begin planning the expansion of the wide area network. The NFS as well as other distributed file systems (the Newcastle Connection and RFS from ATT on UNIX systems) are already running on hardware from many vendors. One would hope that supercomputer vendors will follow the lead and provide such an interface to their machines. At the same time, it seems likely that other distributed applications will be built on top of YDR and RFS. It will be very interesting to see how much of this is actually in place at the Cray User Group meeting a year from now.

** MSDOS is a trademark of Microsoft Corporation.

*** VMS is a trademark of Digital Equipment Corporation.

ACKNOWLEDGMENTS

Thanks to Joe Choy, Ann Cowley, Margaret Drake, Buck Frye, Dave Fulker, and Bob Lackman both for the many conversations on these topics and for the ideas that I've probably incorporated that were originally theirs.

APPENDIX: SCIENTIFIC TASKS

This appendix contains descriptions of tasks performed by scientists and their staff. The focus of the discussion is on areas where computerized automation can lead to significant productivity gains and where centralized resources are needed.

Constructing Scientific Theories

While this part of the scientific process is arguably the most important, it is also least amenable to automation. Some of the outlining tools constructed for business executives (Think Tank is an example) might be useful, but, when you get right down to it, this is mainly a human cerebral process and is not susceptible to much in the way of automation. However, computerization and automation can be quite useful for all the other elements in the list, almost all of which support this function in one way or another.

Teaching

Many university departments are putting together computer laboratories for conducting classes. Others provide or require microcomputers for all their students. The scientists at such institutions would very likely be interested in using some of the same computer systems for accessing archived data or for submitting jobs to supercomputer centers. This means that communications and networking facilities are an important consideration.

Laboratory and Field Experiments

Included in this category are all types of scientific observational experiments. Some examples are simply routine gathering of weather data throughout the world, remote sensing satellite experiments, solar observations from earth or from satellites, laboratory and balloon-born chemistry experiments, etc. Increasingly, scientists are using computers to monitor and control these experiments. Often the experimental apparatus itself includes a microprocessor of some sort. In terms of automation and computerization, the important characteristics of this type of work include the following:

- Some form of local data storage is needed

- In most case, mechanism for moving data to an archival store or to another machine for reduction and analysis is required.

- Real time response is often important for computers receiving experimental data or controlling the experiment.

- In some cases, computers are used to control the experiment interactively. For example, an observing program might be generated on a computer and communicated to a satellite.

- Specialized computer expertise is often needed in the areas of process control, real-time operating systems, communications, expert systems as well as traditional areas.

- Computers used to control experiments are usually specialized in some way. They are almost always associated with a specific project or experiment, hence they are not usually supported by computing centers, but are the responsibility of a particular department, division, project or individual experiment.

- Scientists may want to use the computer that controls their experimental apparatus to perform some of their other scientific tasks. This can put constraints on how some of the other things are done. For example the computer may not be available for data reduction when it is busy recording experimental data from a satellite.

As in most other areas, it's important to be able to move data between machines quickly and conveniently.

Computer Modelling Experiments

While much modelling and simulation can be and is done on computers of modest size and capability, this field is often associated with supercomputing, because many scientific models can tax the computing power and memory of the largest computers. Since this is especially true in the fields of meteorology, oceanography, and astronomy, we will focus on supercomputer modelling here. The important characteristics are:

- High speed computing, fast I/O and large central external memory systems are crucial.

- Facilities for archiving the output of models are needed.

- Access to history or expermental data may be needed to provide a starting point for the model. The ability to store and share this kind of data are important.

- Special expertise in numerical analysis is always required, In many cases, knowledge of particular architectures for parallel and vector processing or using a floating point accelerator is helpful.

- Libraries of mathematical and numerical software are required.

- Program development tools, such as compilers, editors, debuggers, timing profilers, and the like, are needed. In some cases, some of these functions can be distributed to a frontend computer, but access to a good interactive debugger on the modelling computer itself is usually a boon to productivity.

- Special modelling packages may be available for some disciplines.

- In some cases, it can be important to examine the state of a model interactively at various points to determine whether and how it should continue.

- While modelling in the atmospheric sciences is usually very computer-intensive and traditionally has been done at central supercomputer sites, the advent of minisupercomputers (e.g. Alliant and Convex), superfast floating point processors and the like may make it more reasonable for some scientists to trade off some computing speed for local access and control.

- For any individual scientist, the availability of multiple supercomputer sites may have a bearing on where certain tasks are done. For example it may make more sense to get a departmental machine for data analysis rather than relying on a machine at one of the supercomputing centers if you many end up doing your supercomputing at another center.

- Some scientists may want to use the modelling computer to perform some of their other scientific tasks as well. This can put constraints on how some of the other things are done. For example, a supercomputer that is needed for a complicated model may not be ideal for driving interactive graphics devices for analyzing the output of the model.

For many scientists, the supercomputer used for modelling experiments is not suitable or available for performing other tasks such as interactive graphical analysis or technical text processing. The implication is that good communications are required between the supercomputer center and the local computer system used for other tasks.

Data Archiving and Retrieval

Computers are now used routinely to gather data from observational experiments and to generate it directly in numerical experiments. It's not always immediately obvious how these vast collections will be used. Even with careful selection and screening processes, the amount of data that is of potential interest can be overwhelming. Many scientists want access to several archives.

For example, most will need a local database for interactive analysis. In many cases, special group projects will have an archive accessible on a project minicomputer in a university department. Finally there are the massive, centralized repositories that serve entire scientific disciplines. The meteorological archives at NCAR are a good example of the latter.

In addition to the obvious hardware requirements for storing the data, scientists accessing the data would benefit greatly from a convenient cataloging procedure. Commercial database management systems provide a mechanism for storing the catalog of the data, if not the data themselves. Some of the newer database systems, which provide for large data fields, may even be suitable for storing some of the data archives.

For scientists accessing multiple databases, a consistent interface would lessen the time required to learn and relearn the interface to each collection. This area is a prime candidate for collaboration cooperation and perhaps some informal standardization in the case of the large collections within a scientific discipline.

Since the archives reside in many locations, it's important to have up-to-date data communications facilities for transferring the data to the machine where the analysis will be done.

Data Analysis

This can be an extremely labor-intensive task. As such, scientists are using computers more and more to increase productivity in this task. The idea here is to use computer technology to speed up the processes whereby a scientist gains insight and understanding by investigating massive amounts of numerical data.

Some of the more important aspects of this function are:

- Access to data from many sources is important. Scientists may want to compare results of models with experimental data, or current experimental data with archived data. These data may reside at many sites in many different formats.
- Graphics software and hardware are crucial for gaining rapid insight into large data collections.
- Interactive access is important. A scientist may want to decide what data to examine next on the basis of an interesting feature found in the current display. For example, an anomaly in the temperature pattern for a given time and area might warrant looking at the previous day's temperature in the same place or checking out the pressure at the same place and time.
- Software and expertise in numerical analysis and statistics is needed in most cases.
- Specialized software and expertise in signal and image processing may be needed for reduction of certain experimental data.
- Data processing packages may be useful, for example, GENPRO, the CCM Processor, etc.
- "Turnkey" software and hardware systems are available for certain datasets, e.g. McIdas from Wisconsin, DSP from Miami, TAE/GEMPAK/GEMPLT from NASA.
- Since the data to be analyzed may come from many sources, it is often advantageous to use a local divisional, departmental computer or a personal computer or workstation. This approach means that connections to the sources of the data are of added import as are facilities for communicating with collaborators and colleagues at other sites.
- It is especially true that scientists would like to use the data analysis computer for many other tasks. For example it is extremely useful to be able to communicate interesting results to colleagues electronically from the machine where the results are first noticed. Likewise, it is helpful to take numerical data, graphs and pictures and immediately incorporate them into viewgraphs, and papers without having to learn an entirely different system. If scientists can use this machine to access data archives, run models on various supercomputers and control experiments as well, it can make life much simpler.

Reviewing Literature and Publications

The scientist's needs in this area are not that much different from those of other professionals. This is a case where scientists can take advantage of technology developed primarily with business applications in mind. Business applications are typically not written with a supercomputer in mind, so much of the available software for library searches and online bibliographies is written for minicomputers or microcomputers.

Here again, good communications facilities are a key element in gaining electronic access to published information. These facilities include:

- connected computer networks,
- electronic mail,
- video conferencing.

As with many of the other tasks, the scientist can be more productive if the same computer can be used to access published information, to perform data analysis, and to incorporate quotations, data, and graphics into publications.

Preparing Presentations and Publications

While wordprocessing software and hardware abounds, much of it falls short in the demanding area of technical text processing. The ability to include equations in the text is essential for most scientists and greatly limits the number of useful software packages and output devices. Likewise, it is especially important for scientists to incorporate complicated scientific graphics in presentations and publications. This is another area where many of the business-oriented packages fall short. On the other hand the packages that work well for technical text processing and graphics tend to be more complicated and less user-friendly. These are fields in which centralized sources of standardized software, training, consultation, and even output devices can be used very effectively.

Here again it is often the case that scientists would like to do their text processing and graphical presentation preparation on the same computer where their data analysis is done.

Communicating with Colleagues and Staff

Most of the requirements here are similar to those listed under Reviewing Literature and Publications. In this case, the scientist has to be able to send and receive messages among her own workstation and those of other scientists.

The required technology includes:

- connected computer networks,
- electronic mail,
- video conferencing.

CRAY-2 UNICOS Kernel Enhancements

Timothy W. Hoel

Cray Research, Inc.
Mendota Heights, MN

Introduction

This paper will describe the changes to the CRAY-2 UNICOS kernel which have happened during the last year, May 1985 -- May 1986. To help place this year in context, a few highlights from before this period will first be mentioned. After describing this year's changes in detail, some future plans will be discussed.

Background

A single processor CRAY-2 was installed at Mendota Heights in August 1984. We had prepared for its coming by modifying the kernel to run on a CRAY-2 simulator which executed on a CRAY X-MP. Actually, there were two simulators: one for the foreground processor and one for a background processor. We had simulated the foreground code and background code independently, but they first came together on the real hardware. In October 1984 we demonstrated UNICOS on a CRAY-2 running a few simple commands. By November 1984 we were able to transfer files across the HYPERchannel using our own Simple, Effective Protocol (SEP). We also had a very limited batch capability at this time by having a daemon waiting on the CRAY-2 ready to execute any shell script which appeared in a special directory. It was the user's responsibility to include commands to send any output files back to the front end. We added asynchronous I/O (reada/writea) by February 1985. In March 1985 we demonstrated UNICOS executing four separate user processes on a four processor CRAY-2 in Chippewa Falls.

Track Allocation

The Unix® System V operating system allocates disk space sector-at-a-time from a LIFO, push-down stack of available sectors. This mechanism has the virtues that the algorithms and data structures are simple and the disk space is used efficiently, i.e. no more than one sector is wasted per file due to round-off. However, there are some disadvantages to this mechanism as well. Although the push-down stack starts out ordered, it tends to become quite disordered over time as files are randomly created and deleted. Then when a new file is created the sectors allocated to it tend to be scattered, requiring a lot of relatively slow head motion on the disk. This mechanism is quite appropriate in a mini-computer environment where the cpu's are not fast enough to consume the data at disk rates anyway, and further, with many users sharing a few drives it is quite likely that a different user will make an intervening disk request and "steal the heads away". However, Cray computers are quite capable of consuming several streams of disk data so we had to choose a more appropriate allocation mechanism.

The CRAY-2 hardware and firmware are especially good at reading (and writing) tracks of data from the disks. After sensing the

Unix is a registered trademark of AT&T Bell Laboratories rotational position of the disk, it transfers the next sector to the corresponding position in the memory buffer. In this way a track of data can be moved in one revolution plus, on average, one-half sector time of latency.

To take advantage of this capability, large files are allocated and accessed track-at-a-time. A bitmap is used to record available tracks so that as a file grows, a "nearby" track can be allocated to minimize seek time. Since interactive systems typically have many small files, we use a variation of the large block/small block system to avoid wasting excessive amounts of disk space. Files that are eight sectors or less are extended sector-at-a-time from a push-down stack as in standard Unix.

Partitions and Clusters

A Unix "file system" normally resides on a single disk, and since a file is contained within a file system, a single file can be no larger than a single disk drive. We have extended UNICOS so that a file system can be stored on one or more disk drives.

A "partition" is a contiguous group of tracks on a disk drive. In addition to the major/minor device numbers, the special node in /dev for a partition also contains other information describing the partition, including the starting track and number of tracks. At mount time and/or device open time this information is copied to system tables for use by the disk driver. Previously this information was hard-coded into the disk driver which made it inflexible.

A file system resides within a "cluster", which is a group of one or more partitions. The partitions of a cluster need not have the same size or placement, but it is expected that they reside on different physical drives.

Disk I/O

During this past year the DD-29 disk driver has been improved to support CRC recovery. Previously, we depended on retries with head offset positioning which, in practice, was very effective. Also during this past year we have added support for DD-49 disk drives. Currently, this driver does not support CRC recovery, but we are planning to add that capability during the coming year.

We have added "striping" for the swap file as a special case since that is a very important instance where high throughput is necessary.

Networking

We have added support for the TCP / IP protocol family, a widely accepted standard. In addition to the Transmission Control Protocol (TCP) and Internet Protocol (IP), the protocol family includes a number of services which "run on top of" TCP and IP. These services include file transfer (FTP and rcp), interactive connections (TELNET), electronic mail (SMTP), and remote execution. TCP / IP is offered as an option for additional cost. We have also added support for Cray's SCP protocol which will be included in Release 2.0 of UNICOS.

Multitasking

Standard Unix has a system call, named fork, which creates a new process and copies the user address space from the original. To allow a user program to execute concurrently on two or more cpus, we have added a new system call, named tfork, which also creates a new process, but one which shares the same user address space. The CRAY-2 hardware supports semaphores which these "Siamese twins" may use to cooperate.

The Fortran Multitasking Library uses tfork and semaphores to create a higher-level abstraction of tasks, locks, and events. This library interface is the same as that supported on COS and X-MP UNICOS.

Crash

Crash is an interactive utility from System V that can be used to display kernel data structures, either from a live system or from a saved dump file. We have ported crash to work with CRAY-2 UNICOS and found it to be extremely useful for examining the kernel and diagnosing failures.

On-line Diagnostics

There are two ways of running hardware diagnostics with UNICOS. In the first mode UNICOS continues to run in the suspect cpu and the diagnostic program runs as a special process. This mode is most useful when the failure does not affect the operating system (e.g. floating point errors) and the diagnostic needs to read files and/or print messages. To accomplish this, the bitmask defining publicly available cpu's is changed to not include the suspect cpu. Then the per process bitmask of the diagnostic is set to indicate "only run in the suspect cpu".

In the second mode UNICOS is turned off in the suspect cpu(s) (or never turned on), but it runs normally in the other, good cpu(s). A monitor program is run under UNICOS which loads the diagnostic program into its own data space and then makes a special foreground call to start the downed cpu executing the diagnostic just loaded. To prevent the diagnostic from interfering with the operating system, BA and LA are set to limit memory access and the foreground refuses to perform any I/O requests. Almost every off-line diagnostic can be run in this second mode.

Other Enhancements

We ported the accounting commands, e.g. acctcom, and can now produce daily/monthly reports and collect acccounting statistics for billing purposes. The precision of statistics was improved enormously by using the real time clock instead of the sampling method of standard Unix. We have also added new statistics such as I/O wait time and a measure of multitasking effectiveness, cpu time using N cpus.

We added the capability to suspend / resume one or more processes, to set per process limits on cpu time and memory usage, and to set the priority (nice value) of other processes.

Other Activities

Although we are primarily responsible for software development, in fact our group provides a variety of services. We spent a great deal of time assisting in the hardware and software installation of the first three CRAY-2 customers. We also spent a great deal of time writing and reviewing documentation for Release 1.0.

Future

We have several projects underway or planned for the coming year. Users will be able to place files on certain drives and/or stripe files across drives by specifying a bit mask at file creation time. The file system will be made more robust by storing some critical directory and inode information redundantly on two different drives ("shadowing"). The swapping and memory scheduling algorithms will be modified to be more effective for our very large memory machine. We plan to add a recovery mechanism which will allow a running process to be saved in a "dropfile" and then be restarted at some later time.

Early Experiences with the NAS Cray-2

John T. Barton
Manager, High Speed Processor Subsystem
Systems Development Branch

Numerical Aerodynamic Simulation (NAS) Program
M/S 233-1
NASA Ames Research Center
Moffett Field, CA 94035

Introduction

The Numerical Aerodynamic Simulation (NAS) Project is creating a supercomputer center at the NASA / Ames Research Center. Although NAS is composed of a large collection of computers, the main computational engine for the project is a Cray-2. This talk covers early experiences at the NAS project with a low serial number supercomputer. The first topic will be networking, the second will be the performance of the CFT Fortran compiler, and for the last a movie will be shown containing graphic output from early scientific results on the Cray-2.

Some Early History

We first gained access to a Cray-2 computer in April 1985. A quadrant, or uni-processor, version of the machine had been shipped to the Magnetic Fusion Energy Computer Center (MFECC) at the Lawrence Livermore National Laboratory. Some preliminary systems development work was possible on the machine, but the Fortran compiler CFT was missing. Cray Research was able to perform some of the contractual requirements on a full 4 processor Cray-2 machine on June 4, after CRI delivered #2001 to MFE. The initial I/O test ran at just over 500 Mbit/sec streaming to 16 disks for over 20 seconds, and the standard SCILIB matrix multiply (MXM) ran at over 1 Gflops.

The pre-shipment factory trials on our machine (#2002) occured in Chippewa Falls on September 18-19, and it arrived at Ames on September 30. At the time we were rather concerned with the state of CFT. It would, for instance, fail to compile 25 out of a selection of 32 codes. It would compile, run, and correctly execute only 1 code of interest to researchers (INS3D), even though it met the contractual requirements of successfully running vectorized versions of two standard benchmark codes (ARC3D and LES).

Although the machine was not a serial number 1 machine, it was the first shipped with the full complement of 256 Megawords of memory, the first with pseudo-banking, the first which ran the UNI-COS operating system full time, and the first to use the CFT compiler. We had, therefore, the moral equivalent of a serial #1 machine.

Early Networking

The system design for the NAS Processing System Network (NPSN) includes both NSC hyperchannel and ethernet communications on all machines, running the TCP/IP protocol. Since the Cray-2 does not as yet support ethernet, we currently have hyperchannel everywhere, and ethernet to all machines except the Cray-2. Our earliest networking was between the Cray-2 uni-processor at MFE and a front-end VAX 11/750, using SEP, the Simple Effective Protocol which Tim Hoel of CRI wrote. It was elementary, point-to-point, and had no checksumming capability. It provided the necessary connectivity to the uni-processor at MFE. After the delivery of our machine to Ames, SEP was ported to a growing collection of front end machines. We were able to effectively use a VAX

11/780 running System V UNIX (UNIX is a trademark of AT&T). which CRI delivered with the Cray-2. We ported SEP to BSD 4.2 UNIX on another VAX 11/780, to our twin Amdahl 5840 machines (with UTS on them) and to our SGI IRIS workstations. One of the benefits of using the UNIX operating system so widely throughout the NPSN was the ease with which we were able to propagate this useful piece of software.

Our present and future solution for networking is the TCP/IP protocol. This reliable method of communication has been successfully used for years on the ARPAnet. Its packet-based structure makes it possible to construct large networks, which are easily re-configurable. If one has some machines in the network which are on hyperchannel but not ethernet, and others on ethernet but whose hyperchannel functionality may be currently down (or never present), then one may use some of the machines which are on both networks as gateway machines. These gateways route packets from the sending host to the recipient. The current gateway is a bsd 4.2 VAX 11/780.

It is the intent of the NAS project that users not be subjected to a confusing assortment of disparate operating systems throughout the network. The UNIX operating system provides a common user interface on all NPSN machines. In the case of the Amdahls, UTS (a UNIX look-alike) runs as a virtual operating system under VM. On the Cray-2, UNI-COS runs in native mode. The UNIX common interface has been extended to the network by means of the Berkeley R-commands and the ARPAnet FTP and TELNET. FTP, or File Transfer Protocol, provides for copying of files from any system in the network to any other, and TELNET provides for remote access through a login procedure. In both cases a password for the remote machine must be passed at the time of the invocation of the utility. The R-commands rlogin, rcp, and rsh do not require any password between trusted hosts. They provide respectively for remote login, copying of files between local and remote hosts, and the remote execution of a shell script (a file containing commands to the operating system).

We have thus far been able to network all of our computers onto a network of machines using TCP/IP. Any machine is a fully functional front-end to any other. In particular, users can directly log into the Cray-2 interactively from wherever they might be. This fosters the use of appropriate machines for front-end activities, such as editing. We do not currently support a full-screen editor on the Cray-2, and do not plan to. We would consider a distributed editor.

FORTRAN Performance - CFT

There was much concern in the Fall and early Winter that CFT would be a problem. We have since had sufficient experience to draw some conclusions regarding what sort of code will run well on the Cray-2 using CFT. We have seen examples of codes that run quite well, and their characteristics have been noted. Those codes that make heavy use of optimized library subroutines such as FFT or MXM (the latter is a matrix multiply) can capitalize on their speed. A code that is either register intensive or local memory intensive will profit from not having to make frequent accesses to main memory. Although there are only 16K words in each of the 4 local memories, one can direct the compiler to store some quantities there. The most recent version of the compiler (V.70) is sufficiently sophisticated to place all scalars from the current subroutine into local memory, unless they have been passed by that subroutine as parameters. Since scalar references are more time-consuming than vector ones, this should be quite beneficial. Finally, those codes which implement algorithms which fully utilize the large memory of the machine are good candidates. We already have large memory algorithms; the unfactored Navier-Stokes code (which is really just a Newton's method) is such an example. It currently uses over 100 Mwords of memory. Other users are scaling up present codes, or changing their algorithms to do more in-memory storage and less recomputation.

Codes which do poorly on the Cray-2 include those that do scalar computations with frequent fetches from and stores to main memory. Partially vectorized codes, by Amdahl's law, fall in this class. The other major factor to be considered is the stride with with calculations are done. If the skip difference (or stride) between consecutive memory references is a power of 2, then memory bank conflicts can occur because the banks have not had sufficient time to recover between accesses. This is easily remedied by having an odd entry for initial array dimensions. For instance, an array previously dimensioned (128,128) should be reset to (129,128).

To illustrate these points we compared the results of running 13 selected codes on the NAS X-MP/12 and the NAS Cray-2. On the average, the codes ran faster on the X-MP. However, those which had been coded specifically for the Cray-2 ran faster on the Cray-2. The remaining 11 of the 13 were originally coded for the X-MP. The two new codes were MATEST, which implements a new matrix multiplication algorithm of David Bailey and Helaman Ferguson, and PITEST, which does a multiple precision calculation of the transcendental constant PI. MATEST, which makes extensive use of the MXM CRI-supplied SCILIB routine for matrix multiplication, ran at 395 Mflops on a dedicated machine, and 244 Mflops on a loaded system. The difference in performance is due to memory bank contention induced by codes running in the other 3 processors. PITEST ran quickly because of its use of an extremely efficient FFT routine. It achieved 144 Mflops on a loaded system.

We anticipate that the performance gap will narrow between the Cray-2 and the X-MP as the compiler matures on the Cray-2. David Bailey presented results at the last CUG meeting (NAS Kernel Benchmark Results, Fall 1985, Montreal) which show that there was an 83% improvement in X-MP speed on the NAS Kernels between two versions of CFT. Between CFT 1.12 and CFT 1.13 the speed rose from 24 to 44 Mflops for the untuned Kernels. It is reasonable to expect advances in optimization will also occur on CFT on the Cray-2, and that even further improvements will be forthcoming in CFT77.

Our History of CFT

There seem to be rumors that CFT is lacking in both robustness and performance on the Cray-2. These rumors are unfounded. The cause for these feelings may well be the state of CFT when we received our Cray-2. Of the 13 codes mentioned in the paragraph before last, only 3 were running at the time of delivery of the Cray-2 in September. Of these 3, 2 were contractually required. By October (CFT V.28) the number had risen to 5, in November (CFT V.37) it was 8, and during the acceptance test in December (CFT V.49) there were 9 out of 13 codes running correctly. By January (CFT V.59) the preponderance of codes were running (11 out of 13), and in April (CFT V.70) all 13 were running. As early as January we began to get very encouraging reports from early users of the system. One of our remote users at Langley Research Center found that he had to make only 4 attempts, all in the course of a single afternoon, to get his converted Cyber 205 code to compile, run, and produce correct answers on the Cray-2.

Although some of the experience this last Fall and Winter with CFT was a bit trying, the NAS project is now quite pleased with both the performance and robustness of CFT on the Cray-2.

Early Scientific Results

From the time that NAS took delivery of the Cray-2 in September, there has been an increasing level of scientific work on the machine. NAS is chartered to support computational fluid dynamics, and early results were made possible by the fact that the CFD code INS3D, an incompressible Navier-Stokes 3-dimensional code, was one of the first codes to correctly run. This work is part of a project at Ames that is studying the Space Shuttle main engines by simulating the flow of fuel in the engine past the LOX posts which carry liquid oxygen into the combustion chamber. The primary result of interest in the calculation is to determine the pressure on those lox posts, for excessive pressure could lead to failure at high power levels. This project was particularly apt for the Cray-2 because of its use of large memory. The early runs used 5 Mwords of main memory, later cases were 8 Mwords, and future runs will be even larger. These calculations would not have been made had the Cray-2 not been at the Ames Research Center.

While preparing the for the graphical viewing of the results, the scientist, Stuart Rogers, was dismayed at the excessively long time that it took to do the particle tracing calculations for the flow field. The IRIS workstation required about an hour of CPU time for this task. The robustness of the CFT compiler is attested to by the ease with which he was able to port those portions of the graphics package which did the particle tracing. With reliable and rapid file transfer capability, and a supercomputer that took only 5 seconds to do the particle tracing computations, the scientist was able to get more effective work from both his graphics workstation. More significantly, the scientist was able to accomplish more research.

Summary

While it is true that NAS has had its share of early vicissitudes with the Cray-2, our experience to date has been very satisfactory. The plan to have all machines in our network mutually communicating with TCP/IP has been successful in achieving connectivity, and is rapidly becoming robust. The CFT Fortran compiler is now firmly on its feet, producing respectably vectorized and optimized code for a wide variety of input. Finally and most importantly, all of the components of the network, including the graphics workstations, have functioned well enough to produce the scientific results exemplified in the film that has been shown.

The Computation of π to 29,360,000 Decimal Digits As a System Test of the NAS Cray-2

David H. Bailey

Sterling Software, Inc.
NASA Ames Research Center

Abstract

In a recent paper [5], Borwein and Borwein derived a class of algorithms based on the theory of complete elliptic integrals that yield very rapidly convergent approximations to elementary constants. The author has implemented Borweins' quartically convergent algorithm for $1/\pi$, using an advanced prime modulus multi-precision technique, to compute over 29,360,000 digits of the decimal expansion of π. The result was checked by using a different algorithm, also due to the Borweins, that converges quadratically to π. These computations were performed as a system reliability and performance test of the Cray-2 operated by the Numerical Aerodynamic Simulation (NAS) Program at NASA Ames Research Center. The calculations were made possible by the very large main memory of the Cray-2.

Until very recently the largest verified computation of the expansion of π was due to Kanada and Tamura [11] of the University of Tokyo. In 1983 they computed approximately 16 million digits on a Hitachi S-810 computer, of which 10 million were verified in an independent calculation. Late in 1985 Gosper [8] reported computing 17 million digits using a Symbolics workstation. Samples of the author's calculated results have been compared with both of these previous calculations and is in agreement with them.

This paper describes the algorithms and techniques used in this computation, as well as the performance results.

Introduction

The computation of the numerical value of the constant π has been pursued for centuries for a variety of reasons, both practical and theoretical. Certainly a value of π correct to 10 decimal places is sufficient for most "practical" applications. Occasionally there is a need for double-precision or even multi-precision computations involving π and other elementary constants and functions in order to compensate for unusually severe numerical difficulties in an extended computation. However, the author is not aware of even a single case of a "practical" scientific computation that requires the value of π to more than about 100 decimal places.

Beyond immediate practicality, the decimal expansion of π has been of interest to mathematicians, who have still not been able to resolve the question of whether the digits in the expansion of π are "random". In particular, it is widely suspected that the decimal expansions of $\pi, e, \sqrt{2}, \sqrt{2\pi}$, and a host of related mathematical constants all have the property that the limiting frequency of any digit is one tenth, and that the limiting frequency of any n-long string of digits is 10^{-n}. Such a guaranteed property could, for instance, be the basis of a reliable pseudo-random number generator. Unfortunately, this assertion has not been proven in even one instance. Thus there is a continuing interest in performing statistical analyses on the decimal expansions of these numbers to see if there is any irregularity that would suggest this assertion is false.

In recent years, the computation of the expansion of π has assumed the role as a standard test of computer integrity. If even one error occurs in the computation, then the result will almost certainly be completely in error after an initial correct section. On the other hand, if the result of the computation of π to even 100,000 decimal places is correct, then the computer has performed billions of operations without error. For this reason, programs that compute the expansion of π are frequently used by both manufacturers and purchasers of new computer equipment to certify system reliability.

History

The first serious attempt to calculate an accurate value for the constant π was made by the Greeks, who approximated π by computing the areas of equilateral polygons with increasing numbers of sides. More recently, infinite series have been used. In 1671 Gregory discovered the series

$$\pi = 4\tan^{-1}(1) = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots)$$

A more rapidly convergent variation was discovered by Machin in 1706, based on the identity

$$\pi = 16\tan^{-1}(1/5) - 4\tan^{-1}(1/239)$$

In the nearly 300 years since that time, most computations of the value of π, even those performed by computer, have employed some variation of this technique. For instance, a series based on the identity

$$\pi = 24\tan^{-1}(1/8) + 8\tan^{-1}(1/57) + 4\tan^{-1}(1/239)$$

was used in a computation of π to 100,000 decimal digits using an IBM 7090 in 1962 [15]. Readers interested in the history of the computation π are referred to Beckman's entertaining book on the subject [2].

New Algorithms for π

Only very recently have algorithms been discovered that are fundamentally faster than the above techniques. In 1976 Brent [6] and Salamin [14] independently discovered an approximation algorithm based on elliptic integrals that yields quadratic convergence to π. With all of the previous techniques, the number of correct digits increases only linearly with the number of iterations performed. With this new algorithm, each additional iteration of the algorithm approximately *doubles* the number of correct digits. Kanada and Tamura employed this algorithm in 1983 to compute π to over 16 million decimal digits.

More recently, J. M. Borwein and P. B. Borwein [4] discovered an even simpler quadratically convergent algorithm for π, together with similar algorithms for fast computation of all the elementary functions. Their quadratically convergent algorithm for π can be stated as follows: Let $a_0 = \sqrt{2},\ b_0 = 0,\ p_0 = 2 + \sqrt{2}$. Iterate

$$\begin{aligned} a_{k+1} &= \frac{(\sqrt{a_k} + 1/\sqrt{a_k})}{2} \\ b_{k+1} &= \frac{\sqrt{a_k}(1 + b_k)}{a_k + b_k} \\ p_{k+1} &= \frac{p_k b_k (1 + a_{k+1})}{1 + b_{k+1}} \end{aligned}$$

Then p_k converges quadratically to π: successive iterations of this algorithm yield 3, 8, 19, 41, 83, 170, 345, 694, 1392, and 2788 correct digits of the expansion of π. However, it should be noted that this algorithm is not self-correcting for numerical errors, so that all iterations must be performed to full precision. In other words, in a computation of π to 2788 decimal digits using the above algorithm, each of the ten iterations must be performed with more than 2788 digits of precision.

Most recently the Borweins [5] have discovered a general technique for obtaining even higher order convergent algorithms for certain elementary constants. Their quartically convergent algorithm for $1/\pi$ can be stated as follows: Let $a_0 = 6 - 4\sqrt{2}$ and $y_0 = \sqrt{2} - 1$. Iterate

$$y_{k+1} = \frac{1 - (1 - y_k^4)^{1/4}}{1 + (1 - y_k^4)^{1/4}}$$

$$a_{k+1} = a_k(1 + y_{k+1})^4 - 2^{2k+3} y_{k+1}(1 + y_{k+1} + y_{k+1}^2)$$

Then a_k converges quartically to $1/\pi$: each successive iteration approximately *quadruples* the number of correct digits in the result. As in the previous case, each iteration must be performed to at least the level of precision desired for the final result.

Multi-Precision Arithmetic Techniques

A key element of a very high precision computation of this sort is a set of high-performance routines for performing multi-precision arithmetic. A naive approach to multi-precision computation would not only require a prohibitive amount of processing time, but would sharply increase the probability that a hardware error would occur. In addition to employing advanced algorithms for such key operations as multi-precision multiplication, it is imperative that these algorithms be implemented in a style that is conducive for high-speed computation on the computer being used.

The computer used for these computations is the Cray-2, part of the NAS Processing System Network at the NASA Ames Research Center. This computation was performed to test the integrity of the Cray-2 hardware, as well as the Fortran compiler and the Unix-based[1] operating system. The Cray-2 is particularly well suited for this computation because of its very large main memory, which holds $2^{28} = 268,435,456$ words (one word is 64 bits of data). This capacity is more than the combined main memories of all previously delivered Cray computers. With this huge capacity, all data for these computations can be contained entirely within main memory, insuring ease of programming and fast execution.

No attempt was made to employ more than one of the four central processing units in the Cray-2. Thus, at the same time these calculations were being performed, the computer was executing other jobs on the other processors. However, full advantage was taken of the vector operations and vector registers of the system. Considerable care was taken in programming to insure that the multi-precision algorithms were implemented in a style that would admit vector processing. Most key loops were automatically vectorized by the Cray-2 Fortran compiler. For those few that were not automatically vectorized, compiler directives were inserted to force vectorization. As a result of this effort, virtually all arithmetic operations were performed in vector mode, which on the Cray-2 is approximately 20 times faster than scalar mode. Because of the high level of vectorization that was achieved using the Fortran compiler, it was not necessary to use assembly language, non-standard constructs, or library subroutines.

[1]Unix is a trademark of AT&T Bell Laboratories.

A multi-precision number is represented in these computations as an $(n+2)$-long array of floating-point whole numbers. The first cell contains the sign of the number, either 1, -1, or 0 (reserved for an exact zero). The second cell of the array contains the exponent (powers of the radix), and the remaining n cells contain the mantissa. The radix selected for the multi-precision numbers is 10^7. Thus the number 1.23456789 is represented by the array $1, 0, 1, 2345678, 9000000, 0, 0, \cdots, 0$.

A floating-point representation was chosen instead of an integer representation because the hardware of numerical supercomputers such as the Cray-2 is designed for floating-point computation. Indeed, the Cray-2 does not even have full-word integer multiply or divide hardware instructions. Such operations are performed by first converting the operands to floating-point form, using the floating-point unit, and converting the results back to fixed-point (integer) form. A decimal radix was chosen instead of a binary value because multiplications and divisions by powers of two are not performed any faster than normal on the Cray-2 (in vector mode). Since a decimal radix is clearly preferable to a binary radix for program troubleshooting and for input and output, a decimal radix was chosen. The value 10^7 was chosen because it is the largest power of ten that will fit in half of the mantissa of a single word. In this way two of these numbers may be multiplied to obtain the exact product using ordinary single-precision arithmetic.

Multi-precision addition and subtraction are not computationally expensive compared to multiplication, division, and square root extraction. Thus simple algorithms suffice to perform addition and subtraction. The only part of these operations that is not immediately conducive to vector processing is releasing the carrys for the final result. This is because the normal "schoolboy" approach of beginning at the last cell and working forward is a recursive operation. On a vector supercomputer this is better done by starting at the beginning and releasing the carry only one cell back for each cell processed. Unfortunately, it cannot be guaranteed that one application of this process will release all carrys (consider the case of two or more consecutive 9999999's, followed by a number exceeding 10^7). Thus it is necessary to repeat this operation until all carrys have been released (usually one or two additional times). In the rare cases where three applications of this vectorized process are not successful in releasing all carrys, the author's program resorts to the scalar "schoolboy" method.

Provided a fast multi-precision multiplication procedure is available, multi-precision division and square root extraction may be performed economically using Newton's iteration, as follows. Let x_0 and y_0 be initial approximations to the reciprocal of a and to the reciprocal of the square root of a, respectively. Then

$$\begin{aligned} x_{k+1} &= x_k(2 - ax_k) \\ y_{k+1} &= \frac{y_k(3 - ay_k^2)}{2} \end{aligned}$$

both converge quadratically to the desired values. One additional full-precision multiplication yields the quotient and the square root, respectively. What is especially attractive about these algorithms is that the first iteration may be performed using ordinary single-precision arithmetic, and subsequent iterations may be performed using a level of precision that approximately doubles each time. Thus the total cost of computation is only about twice the cost of the final iteration, plus the one additional multiplication. As a result, a multi-precision division costs only about five times as much as a multi-precision multiplication, and a multi-precision square root costs only about seven times as much as a multi-precision multiplication.

Multi-Precision Multiplication

It can be seen from the above that the key component of a high-performance multi-precision arithmetic system is the multiply operation. For modest levels of precision (fewer than about 1000 digits), some variation of the usual "schoolboy" method is sufficient, although care must be taken in the implementation to insure that the operations are vectorizable. Above this level of precision, however, other more sophisticated techniques have a significant advantage. The history of the development of high-performance multiply algorithms will not be reviewed here. The interested reader is referred to Knuth [12]. It will suffice to note that all of the current state-of-the-art techniques derive from the following fact of Fourier analysis: Let $F(x)$ denote the discrete Fourier transform of the sequence $x = (x_0, x_1, x_2, \cdots, x_{N-1})$, and let $F^{-1}(x)$ denote the inverse discrete Fourier transform of x:

$$\begin{aligned} F_k(x) &= \sum_{j=0}^{N-1} x_j \omega^{jk} \\ F_k^{-1}(x) &= \frac{1}{N} \sum_{j=0}^{N-1} x_j \omega^{-jk} \end{aligned}$$

where $\omega = e^{-2\pi i/N}$ is a primitive N-th root of unity. Let $C(x,y)$ denote the convolution of the sequences x and y:

$$C_k(x,y) = \sum_{j=0}^{N-1} x_j y_{k-j}$$

where the subscript $k-j$ is to be interpreted as $k-j+N$ if $k-j$ is negative. Then the "convolution theorem" (see [13], p. 63) states that

$$F[C(x,y)] = F(x) * F(y)$$

or expressed another way

$$C(x,y) = F^{-1}[F(x) * F(y)]$$

This result is applicable to multi-precision multiplication in the following way. Let x and y be n-long representations of two multi-precision numbers (without the sign or exponent words). Extend x and y to length $2n$ by appending n zeroes at the end of each. Then the multi-precision product z of x and y, except for releasing the carrys, can be written as follows:

$$\begin{aligned}
z_0 &= x_0y_0\\
z_1 &= x_0y_1 + x_1y_0\\
z_2 &= x_0y_2 + x_1y_1 + x_2y_0\\
&\cdot\\
&\cdot\\
z_{n-1} &= x_0y_{n-1} + x_1y_{n-2} + \cdots + x_{n-1}y_0\\
&\cdot\\
&\cdot\\
z_{2n-3} &= x_{n-1}y_{n-2} + x_{n-2}y_{n-1}\\
z_{2n-2} &= x_{n-1}y_{n-1}\\
z_{2n-1} &= 0
\end{aligned}$$

It can now be seen that this "multiplication pyramid" is precisely the convolution of the two sequences x and y, where $N = 2n$. The convolution theorem states that the multiplication pyramid can be obtained by performing two forward discrete Fourier transforms, one vector complex multiplication, and one reverse transform, each of length $N = 2n$. Once the resulting complex numbers have been rounded to the nearest integer, the final multi-precision product may be obtained by merely releasing the carrys as described in the section above on addition and subtraction.

The key computational savings here is that the discrete Fourier transform may of course be economically computed using some variation of the "fast Fourier transform" algorithm. It is most convenient to employ the radix two fast Fourier transform since there is a wealth of literature on how to efficiently implement this algorithm (see [1], [7], and [16]). Thus it will be assumed from this point that $N = 2^m$ for some integer m.

One useful "trick" can be employed to further reduce the computational requirement for complex transforms. Note that the input data vectors x and y are purely real. There is a simple procedure (see [7], p. 169) for combining two real vectors into one complex vector, performing one transform, and then recapturing the separate complex transforms. Thus only one forward transform and one reverse transform need to be performed.

One important item has been omitted from the above discussion. If the radix 10^7 is used, then the product of two cells will be in the neighborhood of 10^{14}, and the sum of a large number of these products cannot be represented exactly in the 48-bit mantissa of a Cray-2 floating-point word. In this case the rounding operation at the completion of the transform will not be able to recover the exact whole number result. As a result, for the complex transform method to work correctly, it is necessary to alter the above scheme slightly. The simplest solution is to use the radix 10^6 and to divide all input data into two words with only three digits each. Although this scheme greatly increases the memory space required, it does permit the complex transform method to be used for multi-precision computation up to about one million digits.

Prime Modulus Transforms

Some variation of the above method has been used in almost all high-performance multi-precision computer programs, including the program used by Kanada and Tamura. However, it appears to break down for very high precision computation (beyond about one million digits on the Cray-2), due to the round-off error problem mentioned above. The input data can be further divided into two digits per word or even one digit per word, but only with a substantial increase in run time and in the already excessive memory requirement. Since a principal goal in this computation was to remain totally within the Cray-2 main memory, a somewhat different method was used.

It can readily be seen that the technique of the previous section, including the usage of a fast Fourier transform algorithm, can be applied in any number field in which there exists a primitive N-th root of unity ω. This requirement holds for the field of the integers modulo p, where p is a prime of the form $p = kN + 1$ (see [10], p. 85). One significant advantage of using a prime modulus field instead of the field of complex numbers is that there is no need to worry about round-off error in the results, since all computations are exact.

However, there are some difficulties in using a prime modulus field for the transform operations above. The first is to find a prime p of the form $kN + 1$, where $N = 2^m$. The second is to find a primitive N-th root of unity modulo p. As it turns out, it is not too hard using a computer to find both of these numbers by direct search. Thirdly, one must compute the multiplicative inverse of N modulo p. This can be done using a variation of the Euclidean algorithm from elementary number theory. Note that each of these calculations needs to be performed one time only.

A more troublesome difficulty in using a prime modulus transform is the fact that the final multiplication pyramid results are only recovered modulo p. If p is greater than about 10^{24} then this is not a problem, but the usage of such a large prime would require *quadruple* precision arithmetic operations to be performed in the inner loop of the fast Fourier transform, which would very greatly increase the run time. A simpler and faster approach to the problem is to use two primes, p_1 and p_2, each slightly greater than 10^{12}, and to perform the transform algorithm above using each prime. Then the Chinese remainder theorem may be applied to the results modulo p_1 and p_2 to obtain the results modulo the product p_1p_2. Since p_1p_2 is greater than 10^{24}, these results will be the exact multiplication pyramid numbers. Unfortunately, double precision arithmetic must still be performed in the fast Fourier transform and in the Chinese remainder theorem calculation. However, the whole number format of the input data simplifies these operations, and it is possible to program them in a vectorizable fashion.

Some authors (see [3], p. 90) have suggested using three transforms with three primes p_1, p_2, and p_3, each of which is just smaller than half of the mantissa, and using the Chinese remainder theorem to recover the results modulo $p_1p_2p_3$. In this way double precision operations are completely avoided in the fast Fourier transform. This scheme runs very fast, but unfortunately the largest transform that can be performed on the Cray-2 using this system is $N = 2^{19}$, which corresponds to a maximum precision of about three million digits.

Readers interested in studying about prime modulus number fields, the Euclidean algorithm, or the Chinese remainder theorem are referred to any elementary text on number theory, such as [9] or [10]. Knuth [12] and Borodin [3] also provide excellent information on using these tools for computation.

Computational Results

The author has implemented all three of the above techniques for multi-precision multiplication. The three-prime transform scheme appears to run the fastest, about 35% faster than the complex transform scheme and 70% faster than the two-prime transform scheme. However, the memory requirement of the two-prime scheme is significantly less than either the three-prime or the complex scheme, and the maximum precision level of the two-prime scheme is over three billion decimal digits. Thus the two-prime scheme was selected for the computations of π.

One of the author's computations used twelve iterations of Borweins' quartic algorithm for $1/\pi$, followed by a reciprocal operation, to yield 29,360,128 digits of π. In this computation approximately 12 trillion arithmetic operations were performed. The run took 28 hours of processing time on one of the four Cray-2 central processing units and used 138 million words of main memory. It was started on January 7, 1986 and completed January 9, 1986. The program was not running this entire time – the system was taken down for service several times, and the run was frequently interrupted by other programs. Restarting the computation after a system down was a simple matter since the two key multi-precision number arrays were saved on disk after the completion of each iteration.

This computation was checked using 24 iterations of Borweins' quadratically convergent algorithm for π. This run took 40 hours processing time and 147 million words of main memory. A comparison of these output results with the first run found no discrepancies except for the last 24 digits, a normal truncation error. Thus it can be safely assumed that at least 29,360,000 digits of the final result are correct.

REFERENCES

1. Bailey, D. H., "A Fast Fourier Transform Without Power-of-Two Memory Strides", submited to *SIAM Statistical and Scientific Computing*, 1985.

2. Beckman, P., *A History of Pi*, Golem Press, New York, 1971.

3. Borodin, A., Munro, I., *The Computational Complexity of Algebraic and Numeric Problems*, American Elsevier Publishing Co., New York, 1975.

4. Borwein, J. M., and Borwein, P. B., "The Arithmetic-Geometric Mean and Fast Computation of Elementary Functions", *SIAM Review*, 26 (1984), pp. 351-366.

5. Borwein, J. M., and Borwein, P. B., "Elliptic Integrals and Approximations to Pi", unpublished manuscript, 1985.

6. Brent, R. P., "Fast Multiple-Precision Evaluation of Elementary Functions", *Journal of the Association of Computing Machinery*, 23 (1976), pp. 242-251.

7. Brigham, E. O., *The Fast Fourier Transform*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

8. Gosper, W., private communication.

9. Grosswald, Emil, *Topics from the Theory of Numbers*, Macmillan, NY, 1966.

10. Hardy, G. H., and Wright, E., M., *An Introduction to the Theory of Numbers*, 5th edition, Oxford University Press, London, 1984.

11. Kanada, Y., and Tamura, Y., "Calculation of π to 10,013,395 Decimal Places Based on the Gauss-Legendre Algorithm and Gauss Arctangent Relation", Computer Centre, University of Tokyo, 1983.

12. Knuth, D., *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, Addison-Wesley, Reading, MA, 1981.

13. Papoulis, A., *Signal Analysis*, McGraw-Hill, NY, 1977.

14. Salamin, E., "Computation of π Using Arithmetic-Geometric Mean", *Mathematics of Computation*, 135 (1976), pp. 565-570.

15. Shanks, D., and Wrench, J. W., "Calculation of π to 100,000 Decimals", *Mathematics of Computation*, 16 (1962), pp. 76-79.

16. Swarztrauber, P. N., "FFT Algorithms for Vector Computers", *Parallel Computing*, 1 (1984), pp. 45-64.

USER REQUIREMENTS COMMITTEE REPORT

Stephen Niver

Boeing Computer Services
Seattle, Wa.

The role of the User Requirements Committee (URC) is to act as a focal point for technical requests beteween the Cray User Group (CUG) and Cray Research, Incorporated (CRI). To this end the URC collects requirements from CUG membership and the Special Interest Committees (SICs) reviews the requirements for general applicability and suitability, conducts a ballot of the CUG membership, presents the results to the CUG membership and CRI and recommends to the CUG Board of Directors (BOD) the items to be forwarded to CRI for an official response. In addition to balloting, the URC will, from time to time solicit CRI response to various issues; in these cases the URC will request an official CRI statement of policy or intent. The URC is also responsible for mmaking the ballot results and the official CRI response available to the CUG membership.

Following are the CRI response to the items that were forwarded to CRI from the Winter 1985 ballot.

Job Dependencies

Provide the capability for an end user to define the order in which a given set of jobs will execute, regardless of the order of submission. The job sequence should be alterable by user or system operator; the sequence information should be preserved across system recoveries. It should also be possible to define the sequence or before after all of the affected jobs have been submitted.

RESPONSE: We have no plans to implement a job dependency capability.

Permanent Dataset Protection

The design of the CRI permanent dataset system is powerful but provides a single point of catastrophic failure, the dataset catalogue (DSC). Provide a capability for significantly minimizing the possibility of the DSC being corrupted or, alternatively, minimizing the impact of the destruction of the DSC.

RESPONSE: COS 1.15 STARTUP processing significantly improves DSC error recovery. STARTUP will correct block number errors when reading the DSC and only datasets described by actual bad blocks will be lost. In addition, the COS 1.15 archiving feature provides the capability to recover datasets easily in case of catastrophic failure. After restoration of the backup dataset catalog, datasets can be loaded either as a utility function, or alternatively they can be recalled automatically as they are requested by users.

SCILIB Extensions

When the Cray systems were initially offered, several mathematical libraries were offered for use in attracting new programs and providing a degree of optimization. Use of and demand for these capabilities have grown to the extent that their availability is now a requirement. Please extend your existing SCILIB product to include vectored versions of the more complex mathematical functions such as those included in EISPACK and LINPACK.

RESPONSE: The existing SCILIB product, which includes LINPACK and EISPACK, comprises almost 250 mathematical subroutines. Of these, approximately 90 have been optimized for the CRAY 1-S and X-MP and 70 for the CRAY-2. We are continuing to optimize routines in LINPACK and EISPACK, along with the others. However, since the number of routines in these two libraries is rather large (approximately 160) we are concentrating our efforts on a few of the frequently used subroutines. We would appreciate a prioritized list of routines in LINPACK and EISPACK to be optimized from the CUG Special Interest Group. (I have committed to providing to the software tools SIG a list of sites that voted for this item on the ballot.)

In addition to the items from the ballot forwarded to CRI, CUG requested a CRI statement of policy on these items. Following are the items and the CRI response:

(1) Provide customers at each site with interactive access ot "the" SPR database. The current method of relying upon an intermediate party (local CRI analyst) or a local copy of the database is not satisfactory. Among the capabilities desired are the ability to list SPRs by many different criteria and keywords, list suggested corrective code, append comments, and submit SPRs.

RESPONSE: CRI does not plan to provide direct access to its SPR database by other than Cray employees. A new SPR database has recently been created to provide more access capabilities to SPR information.

The new SPR database, implemented with ORACLE, a relational database system, provides a number of enhancements including: searching by various criteria (including keywords and text), field entry of SPR's, fields for corrective code and test cases, user definable reports, and expandable descriptions (to append comments). For more detailed information or other questions contact your local Cray site analysts. The new database will evolve over time to more closely meet the needs of the users. Your comments and suggestions are welcomed.

(2) While large computer memories are certainly the trend, there are currently a large number of CRI systems installed with significantly smaller memory. To enable these sites to continue to function effectively, CRI should adopt a policy of containing the memory growth of system utilities such as CFT, UPDATE, CAL. To support both future large memories and existing smaller ones, perhaps an installation parameter should determine whether the utility uses overlays or expands.

RESPONSE: CRI is sensitive to the increasing size of system utilities and is making efforts to make them as small as possible. Unfortunately, in order to develop utilities which produce better code or provide better performance, it is likely that these packages will continue to grow. Frequently, the size is a function of required data space not the actual utility code. Although we will try to keep utilities small, we cannot guarantee that there will be no further growth in their size.

(3) Upward compatibility is an important issue to all Cray customers. Source compatibility is important, but it is equally unrealistic to expect users to recreate their binaries for each system upgrade. For customers who have regulated applications, the cost is even greater. A major reason for the lack of compatibility is the changes to system tables that reside in a user job's field length. Increases in table length or moving fields usually invalidates older versions and requires that new binaries be created. With some foresight and critical review of changes, it should be possible for CRI to make changes to tables so that existing binaries are unaffected.

RESPONSE: CRI appreciates the difficulty associated with changes to system tables within the user's field length and does not intend to unnecessarily make changes with these tables. Changes to these tables should only be made when necessitated for reasons of performance or as unavoidable to implement a new feature.

The Board of Directors disagrees with the CRI response to item 1 (provide customers with direct access to the SPR database) and will request that CRI reconsider their position.

Table one (below) summarizes the results of the winter 1985 balloting. The items above the dashed line are being forwarded to CRI for response. Those items below the solid line are being dropped. The items between the lines will be forwarded to the next ballot, unless this was the third ballot that they appeared on (in which case they will be dropped).

SPRING 1986 CUG USER REQUIREMENT SURVEY RESULTS

RESPONSES SORTED BY TOTAL POINTS

FEATURE TITLE	TOTAL POINTS	PERCENT POINTS	AVERAGE RESP	NUMB RESP
CONTROL STATIONS	337.0	9.1	17.7	19
IMPROVED INTERACTIVE	330.0	8.9	27.5	12
FREEZE/THAW	326.0	8.8	17.2	19
ADA	308.0	8.3	51.3	6
BETTER OPERATOR VISIBILITY FOLLOWING INT	304.0	8.2	17.9	17
FASTER ROLL-IN AND ROLL-OUT	281.0	7.6	17.6	16
POSITION OF LOGFILE ON OUTPUT	280.0	7.6	15.6	18
ON-THE-FLY-DUMPS	264.0	7.1	13.2	20
JCL MULTI-TASKING	205.0	5.5	14.6	14
MULTI-LEVEL SECURITY	197.0	5.3	19.7	10
ENHANCED PERMANENT DATASET MAINTENANCE	148.0	4.0	12.3	12
PROGRAM-RELATED ACCOUNTING	148.0	4.0	9.2	16
CRAY-TO-CRAY COMMUNICATIONS	116.0	3.1	16.6	7
IDENTITY OF JOBS INDEPENDENT OF STATION	115.0	3.1	9.6	12
USER INTERFACE FOR STATION MANAGER	101.0	2.7	10.1	10
BDT INTERFACE FOR FETCH/ACQUIRE	88.0	2.4	11.0	8
PSEUDO-STRIPING	54.0	1.5	6.7	8
UNASSIGNED	50.0	1.4	50.0	1
SHADOW VERSIONS OF PERMANENT DATASETS	48.0	1.3	9.6	5

TABLE 1

SHORT PAPERS

PERFORMANCE OF DD-29 VERSUS DD-39

Mike Ess

Mobil Oil Exploration & Production Services

Dallas, Texas

At our site we recently moved from a Cray-1M with 16 DD-29s to a MOS X-MP with 8 DD-39s. The new disks are better on paper (see figure 1) so we expected an improvement in throughput. But the I/O wait time for jobs in batch now takes longer than before. In defining this problem we have had trouble because we didn't expect performance to go down. We have had to dig up old listings to see just what the I/O performance had been on our old system. Then because that system was gone, we tried to reconstruct the old jobs on the new system. Of course, there have been a lot of changes besides the DD-29s that could not be filtered out. The whole experience brought home the need for tools to measure the system performance in batch. At our site, like most every other site, more than 90% of the wall clock time is spent running batch. And the heavy batch usage from 8 AM to 5 PM is when most of our payroll is waiting for results. This lack of tools to measure performance in batch is a serious omission. Most likely the current benchmarks or the situation in Mendota Heights has more effect on the changes to the operating system than any results of a simulated batch environment. The benchmark results in dedicated mode can be worlds apart from the real world in batch. And the batch environment in Mendota Heights is not a production site.

As an example of the need for such a tool let me explain one reason that might account for the increase in I/O wait times from our old system to our new system. I'm not saying that this is the cause of our problem, but only that there is a need for tools to resolve this possibility. As can be seen from figure 1, we can see that the DD-39s are two times bigger, and except for maximum transfer rate, they should be two times faster. The 8 DD-39s would have the same volume as the 16 DD-29s, and given that the DD-39 characteristics are better, it seemed a reasonable way to go. One hardware difference between the systems is that the number of channels has decreased. This decrease in the number of channels and an increase in channel activity is what I speculate is the cause of increased I/O wait times.

On the hardware side there are just fewer wires connecting 150 million word units than there were on the old system. Exactly half as many channels now service the same volume of storage. There are two reasons why the number of requests per 150 million words has increased on our new system. One is that the DD-39s contain 3 spindles per drive and an I/O request is made per drive rather than per device. On the DD-29 there would be two drives handling requests of the 150 million words so the new system has 50% more requests per 150 million words of disk. The last two reasons combine to say that there is just more contention for the disks. The other hardware consideration is that the X-MP has a faster CPU than the 1M, so we now expect more I/O requests per wall clock second. Also because memory increased from 2 million words to 4 million words there are now more jobs in memory requesting service from the disks.

We are gradually moving our seismic workload from the Cybers to the Cray and this accounts for a continual increase in I/O on our system. Seismic processing has always been I/O intensive because of the volumes of data that must be used to make up for inaccuracies in the data. Most processing occurs on a trace by trace basis, where a trace is a few sectors long. This means a lot of relatively small I/O operations. Another characteristic is that this trace by trace processing was well suited to small memory machines but now that 2D and 3D techniques are becoming more popular, rethinking has to be done about the tradeoffs between large memory usage and I/O. But any change from the existing software must first be justified with results from tools measuring an improvement in performance. An given how a production site runs, the tool must show an improvement in batch mode.

In conclusion, I wish to point out that batch performance is more important than dedicated performance on Cray machines. This is simply because more Crays run longer in batch than in dedicated mode. So even though batch performance is hard to measure that doesn't mean we shouldn't try to measure it. Let's treat dedicated mode results as interesting but not very useful for how Crays are actually used.

	DD-29	DD-39
DRIVES / DEVICE	1	3
WORDS / SECTOR	512	512
SECTORS / TRACK	18	24
TRACKS / CYLINDER	10	5
CYLINDERS / SPINDLE	822	840
TOTAL SECTORS	147,960	101,040 303,120
TOTAL DATA WORDS (MEGAWORDS)	75.7	51.7 155.2
ROTATIONAL SPEED	3600	4000
MINIMUM SEEK TIME	15 ms	5.5 ms
AVERAGE SEEK TIME	50 ms	18 ms
MAXIMUM SEEK TIME	80 ms	35 ms
MAXIMUM TRANSFER RATE (MEGABITS / SECOND)	38.8	52.4

FIGURE 1. DISK CHARACTERISTICS

A BAD EXPERIENCE WITH CFT 1.14

Chris Lazou

University of London

London, England

THE ULCC ENVIRONMENT

The University of London Computer Centre provides large-scale computer services to members of the academic community throughout Britain. These services are based on a Cray-1S/1000 running COS 1.12 / CFT 1.14 BF3 and an Amdahl 470V/8 running MVS SP1.3. Access to the centre is provided over X25 - based wide - area networks in conformity with the ISO model of Open Systems Interconnection (OSI).

The user community consists of postgraduates and university teachers, and totals over 6500 accounts. About 1800 of these accounts are Cray-1S users. Most of the users have to submit their work for a "peer review", to establish whether their work warrants a large-scale computer, before they are allowed to use the Cray-1S. This rather small system is overloaded and our users' requirements are one to two orders of magnitude larger than the computational capacity of the Cray-1S. The work simulated on the Cray-1S at ULCC spans the complete range of academic disciplines from physical to biological sciences on to humanities.

CFT VERSIONS

Apart from CFT 1.14 BF3 residing in the system, we also have CFT 1.11 (old calling sequence), CFT 1.11 (new calling sequence), CFT 1.13 BF2, CFT 1.14 BF2X, CFT 1.14 BF4, and prerelease CFT 1.15 on permanent data sets. Indeed many of the other bug fix versions are also there, which gives you an indication of the inherent instability of CFT as a product. In addition to the Cray Products, we support the Cray Library, the mathematical libraries NAG and IMSL, and some 35 packages and other libraries including graphics.

The size of user programs run on the system, range from small development jobs, to large (several hundred thousand lines of Fortran statements) production jobs, partitioned to use as much of the Cray resources available. Figure 1 shows a typical job.

MIGRATION PATH

As a matter of policy ULCC plans to effect upgrades during the summer when University teachers are free from undergraduate teaching duties. Since our user population is spread around the country, we have adopted the following migration path whenever we wish to upgrade to a new version:

1. Document and distribute any external user changes, noting their possible impact on running programs to the user community.
2. Place CFT and associated products on permanent data sets and provide a procedure to access them.
3. Encourage application programmers at ULCC and the users at large to try new versions of CFT.
4. Generate new libraries on permanent data sets for users to access on a trial basis.
5. A stringent quality assurance exercise is initiated with the aim of assuring that all previous production programs still function correctly with the new versions (an impossible task with Cray software).

PROBLEMS ENCOUNTERED DURING MIGRATION

Once the user community started using CFT 1.14 the problems began to pop out of the woodwork. Our Cray analyst verified and submitted on ULCC's behalf, 11 critical, 5 major, and 2 minor SPRs. There are 3 more known problems which are currently under investigation, not as yet isolated enough to establish whether we have to issue new SPRs for them. Figure 2 shows a list of CFT 1.14 errors encountered. In addition there were 6 errors in the mathematical library $ARLIB. These consisted of errors in (single and double precision) vector SQRT, in double precision vector multiply, and triple precision addition. On several occasions errors were not reported if it was known that the error was fixed at a higher bugfix level. At present another error with complex arithmetic is suspected at CFT 1.14 BF4.

The problems encountered were mainly due to the CFT compiler generating wrong code or the functions in the $ARLIB library having been "speeded up" by changing the algorithms, but with scant respect to accuracy. These problems were detected in large codes such as the LUSAS package (55K lines of code), a computational chemistry program (350K lines of code), a crystallography package, econometrics, GAUSSIAN 82, and the NAG tests. Figure 3 shows the spread of problems in CFT 1.14 and the instability of subsequent versions.

REMEMDIES

With such spread of problems encountered at CFT 1.14, ULCC was unable to upgrade last summer, but eventually upgraded in February 1986. Another problem which may be local to European sites, is that the response to critical SPRs by Cray Research is very slow. Even when code has been developed to solve the critical problem it is often not available to us for several weeks rather than days.

RECOMMENDATIONS

1. CRI should do more testing before releasing its products, if it wishes to preserve the confidence of the user community in their worthiness.

2. CRI should consider providing a mechanism for access of all current SPRs by all sites to enable installations to ascertain whether a problem they are hitting has previously been reported. This has the added advantage, for installation analysts, of providing material hints to assist them when trying to isolate problems in large systems. Some of these problems take days to isolate and any reduction of this unnecessary cost would be appreciated.

3. CRI should consider publishing any changes to algorithms calculating floating point numbers as results from mathematical functions and should try to conform to either IEEE or other suitable standards where available.

4. CRI must do better as far as CFT is concerned if it wishes to keep ahead of its competitors in this field.

- STATISTICS: 594 LOGFILE STATEMENTS
  ACCESS, CFT, LDR, SAVE, ETC.
- 29 SEPARATE CFT COMPILATIONS
- 2 CAL COMPILATIONS
- 107,407 LINES OF FORTRAN
- 95.84 SECONDS TO COMPILE
- IN ADDITION IT CALLS ON PRECOMPILED LIBRARIES AND CREATES AN OVERLAYED ABSOLUTE PROGRAM
- PRODUCTION RUNS ARE THEN DONE WITH DIFFERENT SETS OF DATA
- INTERMEDIATE RESULTS ARE STORED AND PICKED UP AGAIN FOR DIFFERENT RUNS

ENQUIRY:
MY RESULTS ARE NOW WRONG AT THE 6TH DECIMAL POINT WITH NEW VERSION OF CFT?

FIGURE 1. EXAMPLE OF PROGRAM COMPLEXITY

CRAY SPR NO.	CFT VERSION (DATE)	PROBLEM	ANSWER LEVEL
12055	1.14 (4/85)	BAD CODE GENERATED FOR DO LOOP	1.14 (6/85)
12179	1.14 (4/85)	INEFFECTIVE CODE GENERATED FOR MULTIPLE DO LOOPS	1.15 (4/86)
12387	1.14 (5/85)	BAD CODE GENERATED USING T-REGISTERS	1.14 BF2 (9/85)
12400	1.14 (6/85)	BAD CODE WITH LARGE CONSTANT IN DO LOOP	1.14 BF2 (8/85)
12764	1.14	BAD CODE USING WRONG T-REGISTER	1.14 BF3
13007	1.14	BAD CODE GENERATED USING BUFFER IN	-
13303	1.14 BF2 (9/85)	ASSOCIATION OF ENTITIES	10/85
14193	1.14 BF4 (1/86)	CFT FAILS TO PRODUCE CODE	-
14993	1.14 BF4 (3/86)	BAD REGISTER USAGE IN LOOP	-
-	1.14 BF3 (4/86)	B-REGISTER NOT SET UP	-

FIGURE 2. LIST OF PROBLEMS WITH CFT 1.14 AT ULCC

INSTITUTION	CFT 1.13	CFT 1.14 BF2X	CFT 1.14 BF3	CFT 1.14 BF4	CFT 1.15 PRE-RELEASE
KING'S	(OK)	(OK)	X	X	X
IC	(OK)	X	X	(OK)	(OK)
YORK	(OK)	X	X	X	X (USER)
BIRKBECK	(OK)	X	X	X	-
READING	(OK)	(OK)	X	X	-
ULCC	(OK)	X	X	X	X
ULCC	(OK)	(OK)	X	X	-
CAMBRIDGE	(OK)	X	X	(OK)	(OK)
ULCC	(OK)	(OK)	(OK)	(OK)	-

FIGURE 3. MATRIX OF PROBLEMS WITH CFT 1.14 AT ULCC

BATCH JOBS ON UNICOS

Clay Andreason

Cray Research, Inc.

Mendota Heights, Minnesota

WHAT IS NQS?

First I'd like to say that the version of NQS that I will be describing will be available in release 2 of UNICOS.

The Network Queueing System is being developed by a company called Sterling Software under a contract with NASA.

It WILL be supported under UNICOS by Cray Research.

It is based on the Multiple Device and Queueing System written at Ballistics Research Lab.

NQS is designed to allow users to submit, terminate, monitor, and control their own batch jobs.

NQS maintains several batch queues as configured by the system administrator, selects and schedules batch jobs for execution, optionally notifies users of job start and termination, and returns the output to the point of submission.

NQS consists of several parts. They are: nqsdaemon, netdaemon, qmgr, qsub, qstat, and qdel. Figure 1 depicts the general flow of a batch job under the control of NQS.

Nqsdaemon

The NQS daemon spends most of its time waiting for requests to perform functions. Its jobs is to control all queues and their contents. This includes the following functions:

- o Creating, modifying, and deleting queues.
- o Assigning of sequence numbers to requests (jobs).
- o Entering requests into the queues.
- o Moving requests within and between queues.
- o Updating memory and disk resident queue structures.
- o Spawning requests from queues.
- o Removing completed requests.

Netdaemon

The network daemon is optionally started when NQS is started. Its function is to wait for requests sent from other machines. It then maps the remote user id into a local user id and submits the request to the NQS daemon.

Qmgr

The qmgr program is the operator's interface to NQS. It cracks commands, formats them and writes them to the nqsdaemon. It then waits for a reply and tells the operator of the success or failure of the command. Qmgr also provides extensive help facilities for all commands.

Qstat

The qstat program provides status displays of NQS queues and their contents. It provides status information by queue name, request name, or sequence number.

Qdel

The qdel program provides a means to delete requests from queues or to signal all of the process associated with a request.

I'll go into more detail on all of these later.

Queues may be created with a number of attributes. These include:

- o Users allowed in the queue.
- o Groups allowed in the queue.
- o Default/maximum time limit.

- Default/maximum memory limit.
- Default/maximum nice value.
- Priority of queue.
- Maximum number of requests running in the queue at one time (run limit).

These attributes are used to determine which queue each request belongs in.

Any of these attributes may be changed at any time via the qmgr program.

Normally NQS will initiate all requests in a queue up to the run limit for the queue.

A new request will be initiated in a queue whenever one completes in that queue providing that run limits will not be exceeded.

NQS also provides for the grouping of queues into a queue complex.

A run limit may be specified for a queue complex.

Figure 2 is a sample queue configuration display.

Qsub Command

The qsub command is used to submit the script file along with any desired request parameters to the NQS batch subsystem.

Parameters specified on the qsub command override any in the script file.

Qsub validates all of the request parameters and builds a control file for the request.

The NQS daemon is then notified.

Important parameters are:

-a time	Time after which to run.
-j job-name	Job name. (Default is the script file name)
-lM mem-limit	Limit of memory size per job.
-ln nice-limit	Set a nice value for all processes in a job.
-lT cpu-time	Limit of CPU time per job.
-mb	Send mail at beginning of job.
-me	Send mail at end of job.
-nr	Declare the request non-restartable.

Figure 3 shows all the qsub parameters.

WHAT IS A BATCH JOB?

There are many similarities and differences between COS and UNICOS batch jobs. Figure 4 summarizes the differences, which are elaborated below.

Some of the differences are obvious in the sample UNICOS and COS jobs in figures 5 and 6.

- Under UNICOS, the data for a command immediately follows the command. COS normally tacks data files onto the end of the job.
- A job accounting report at the end of your job is available under UNICOS, but only if you ask for it.

One major difference between COS and UNICOS is the user environment provided by UNICOS.

- Every job or interactive session begins by executing a file called ".profile" in the user's home directory. This allows a user to have all kinds of things done at every login.
- This can include executing any UNICOS commands and the setting of environment variables. Environment variables are similar to the symbolic variables in COS JCL, but more powerful because they are available to every program instead of just the command line interpreter.

Another difference that comes to mind is the lack of local or temporary files under UNICOS. This has some interesting side effects.

- Two copies of the same job running in the same directory may easily interfere with each other.
- Because all files exist until explicitly removed, there can be a lot of garbage files staying around that used to disappear under COS when the job terminated.

Fortunately the flexibility of UNICOS provides some solutions to the problem.

- It is possible to set up the user's environment so that when a job is started, a unique directory is created and the job begins executing there. This means that each job has a fresh slate to start from. A job writing to file X won't interfere with another copy of the job writing to file X.

- o It is also possible to set things up so that the directory will be automatically removed when the job terminates. Sounds like COS temporary files, doesn't it? The bonus is that you could also login interactively and look around at the job's files if you wanted to.

One major difference that may cause some problems for COS users is that the command line interpreter (known as the shell) under UNICOS essentially runs in a NO ABORT mode. This means:

- o The shell does not automatically terminate the job when there is an error in some command. The job must include an error check if desired.
- o When each UNICOS process terminates it may (but is not required to) return an exit status to its parent.

Qstat Command

Figure 7 is an example of the output from the qstat command.

The default display is one line for each of your own jobs. Jobs owned by others are indicated but not displayed.

You may ask for status of a particular job or jobs, and may request an extended status.

OPERATOR CONTROL

Operator control is accomplished via the qmgr program. It has the following commands (summarized in figure 8).

Abort Queue

All requests in the named queue that are currently running are aborted as follows. A SIGTERM signal is sent to each process of each request presently running in the named queue. After a specified number of seconds of real time have elapsed, a SIGKILL signal is sent to all remaining processes for each request running in the named queue. All requests aborted by this command are lost.

Create Batch_queue

Define a batch queue with a specified inter-queue priority. If pipeonly is specified, then requests may enter this queue only if their source is a pipe queue. The specification of a run_limit sets a ceiling on the maximum number of requests allowed to run in the batch queue at any given time.

Create Pipe_queue

Define a pipe queue and associate it with a server. This is done by specifying an absolute path name to the program binary server and any arguments required by the program. A list of one or more destination queues that requests from this pipe queue may be sent to is specified.

Delete Queue

The specified queue is deleted. To delete a queue, no requests may be present in the queue and the queue MUST be disabled (see "Disable Queue" below).

Delete Request

Delete the specified request(s). This command can delete both running and non-running requests. If a request is running, then all processes of the request are sent a SIGKILL signal.

Disable Queue

Prevent any more requests from being placed in this queue.

Enable Queue

If the queue is already enabled, then this is a no-op. Otherwise, the queue is enabled to accept new requests.

Help

Get help information. Help without an argument displays information about what commands are available. Help with an argument displays more detailed information about that command. The command may be partially specified as long as it is unique. The more completely specified your help request, the more detailed the information that you will receive.

Purge Queue

All queued requests are purged (dropped) from the queue and are irretrievably lost. Running requests in the queue are allowed to complete.

Set Nice_value

Set a best nice-value for a batch queue, against which the nice-value for a request may be compared. If a request already in the queue has asked for treatment more favorable than the new nice-value, then it will be given a grandfather clause. A request specifying a nice-value may only enter a batch queue if the queue's nice value is numerically less than (more willing to allow access to the CPU) or equal to the request's nice value. A nice-value is an integer preceded by an optional negative sign.

Set Per_Request Cpu_limit

Set a per-request maximum CPU time limit for a batch queue against which the per-request maximum CPU time limit for a request may be compared. If the local host does not support per-request CPU time limits, then this command will report an error. If the local host does support per-request CPU time limits, then every batch queue on the local host will have a per-request maximum CPU time limit associated with it at all times. If a request already in the queue has asked for more than the new limit, then it will be given a grandfather clause. A request specifying a per-request maximum CPU time limit may only enter a batch queue if the queue's limit is greater than or equal to the request's limit.

Set Run_limit

Change the run-limit of an NQS batch or pipe queue. The run-limit determines the maximum number of requests that will be allowed to run in the queue at any given time.

Show Queue

Display the status of the specified queue(s). A standard amount of information for all queues is shown if no queue names are given. If one or more queues are named, then the standard information and the queued request ordering are displayed for the named queues.

Shutdown

Shutdown all moving parts of NQS on the local host. A SIGTERM signal is sent to each process of each request presently running. After the specified number of seconds of real time have elapsed, a SIGKILL signal is sent to all remaining processes for each request. Unlike Abort Queue, Shutdown requeues all of the requests it kills.

Start Queue

If the queue is already started, then nothing happens. Otherwise, the queue is started and requests in the queue are eligible for selection.

Stop Queue

Any requests in the queue that are currently running are allowed to complete. All other requests are "frozen" in the queue. New requests can still be submitted to the queue, but will be "frozen" like the other requests in the queue.

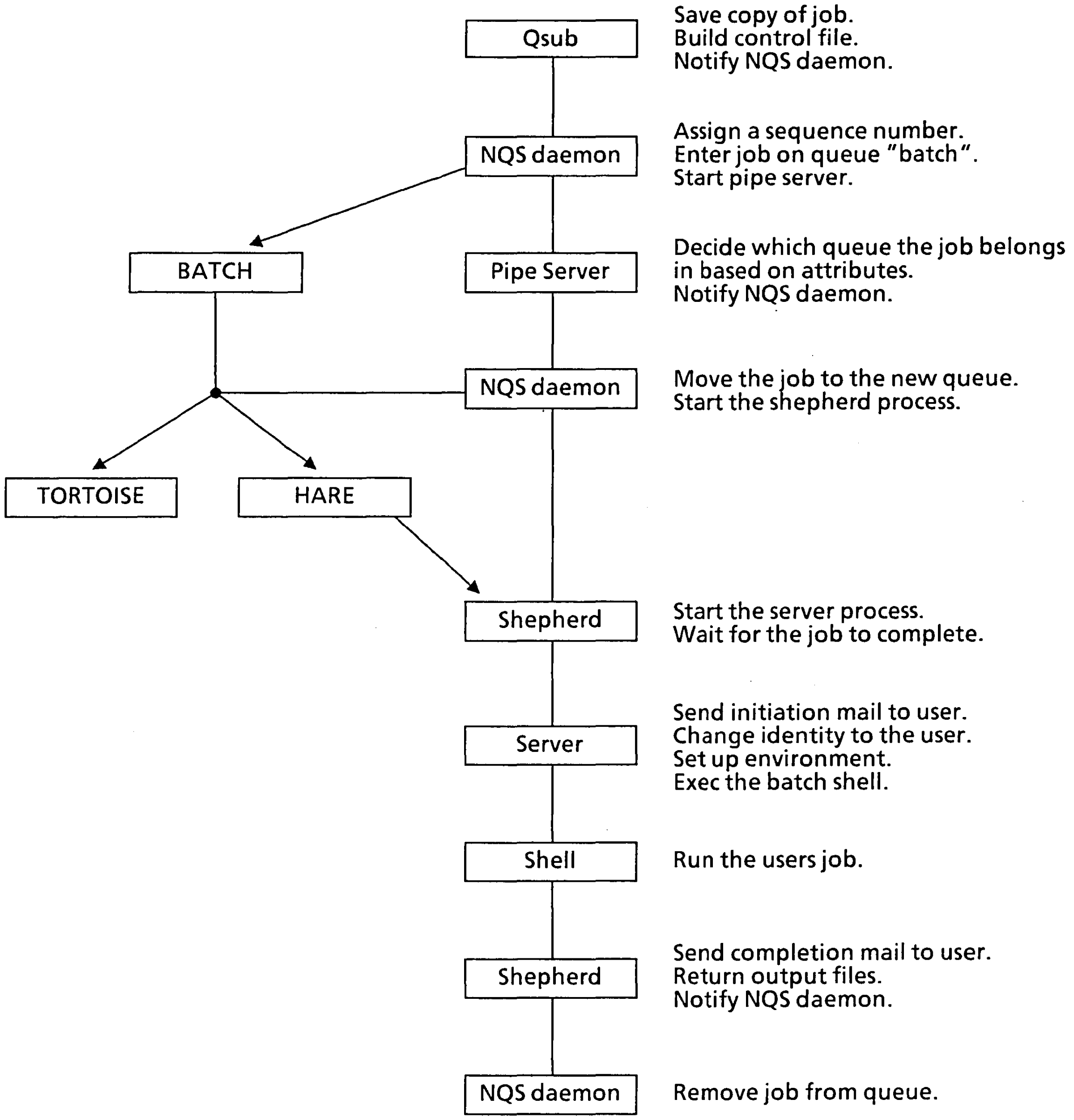

Qsub
Save copy of job.
Build control file.
Notify NQS daemon.
NQS daemon
Assign a sequence number.
Enter job on queue "batch".
Start pipe server.
BATCH
Pipe Server
Decide which queue the job belongs in based on attributes.
Notify NQS daemon.
NQS daemon
Move the job to the new queue.
Start the shepherd process.
TORTOISE
HARE
Shepherd
Start the server process.
Wait for the job to complete.
Server
Send initiation mail to user.
Change identity to the user.
Set up environment.
Exec the batch shell.
Shell
Run the users job.
Shepherd
Send completion mail to user.
Return output files.
Notify NQS daemon.
NQS daemon
Remove job from queue.

Figure 1. NQS Job Flow

Queue	Limit	Priority	Tlimit	Mlimit	Nice	Groups	Users
batch	2	200	---	---	-20	---	---
FAST1	5	90	20	1M	-10	G1	---
NORM1	5	70	100	4M	0	G1	---
BIG1	3	50	100000	20M	10	G1	---
FAST2	5	90	20	1M	-10	G2	---
NORM2	5	70	100	4M	0	G2	---
BIG2	3	50	100000	20M	10	G2	---
HUGE	2	50	---	100M	20	---	sam, joe, sue
ZIP	5	30	1000	8M	-30	bang	---

Queue-complex	Limit
FAST1, NORM1, BIG1	10
FAST2, NORMA2, BIG2	10
BIG1, BIG2	4

Figure 2. Sample Queue Configuration

qsub [parameters] file

The qsub command is used to submit the script file along with any desired request parameters to the NQS batch subsystem.

parameters:	-a time	Time after which to run.
	-e stderr-file	Specify a standard error file name.
	-eo	Combine standard error and standard output.
	-j job-name	Job name. (Default is script file name)
	-ke	Keep stderr file on machine where run.
	-ko	Keep stdout file on machine where run.
	-lm mem-limit	Limit of memory size per-process.
	-lM mem-limit	Limit of memory size per-job.
	-ln nice-limit	Set a nice value for all processes in job.
	-lt cpu-time	Limit of CPU time per-process.
	-lT cpu-time	Limit of CPU time per-job.
	-mb	Send mail at beginning of job.
	-me	Send mail at end of job.
	-mu user	Send mail to another user.
	-nr	Declare the request non-restartable.
	-o stdout-file	Specify a standard output file name.
	-p priority	Queueing priority.
	-q queue	Queue to submit to.
	-s shell-name	Run the request with a different shell.
	-z	Silent submit. No informational message.

Job file:

```
# All commands to NQS are optional, but must appear first.
#@$-j job-name
#@$-lT 10
#@$-me
#@$
...
(shell commands)
...
```

Figure 3. Submitting a Batch Job Request

- UNICOS commands are normally not echoed, but can be.
- Most UNICOS commands give no feedback when successful.
  (No news is good news)
- UNICOS stderr and stdout files are kept separate.
  COS $OUT and $LOG are concatendated.
- No temporary files in UNICOS.
- Shell runs in NO ABORT mode.

Figure 4. UNICOS / COS Differences

```
#
#               UNICOS job.
#
#                               # NQS commands
#               @$-j test       # Jobname is test
#               @$-eo           # Combine standard error and output
#               @$-lt 10        # Time limit of 10 seconds per process
#               @$              # End of NQS commands

set -x                          # Echo commands
jad                             # Start job accounting daemon
cat > f.f << EOF                # Copy input to file f.f
            program test
            integer a(100)
            do 10 i = 1, 100
              a(i) = i
      10    continue
            end
EOF
cft f.f                         # Compile program f.f
ldr f.o                         # Load relocatable f.o as a.out
a.out                           # Execute program a.out
rm f.f f.o a.out                # Remove files
jar -chsf                       # Ask for job accounting


*
*               COS job.
*
JOB, JN=TEST, T=10.
COPYF, O=F.                     . Copy program to file F
CFT, I=F, L=FL.                 . Compile program F
LDR, NX, AB=AOUT.               . Load relocatable $BLD as AOUT
AOUT.                           . Execute program AOUT
/EOF
            program test
            integer a(100)
            do 10 i = 1, 100
              a(i) = i
      10    continue
            end
```

Figure 5. Sample Batch Job Structure, UNICOS vs. COS

```
+ jad
+ cat
          program test
          integer a(100)
          do 10 i = 1, 100
            a(i) = i
     10   continue
          end
+ cft f.f
compiling --- f.f using compiler /cft2/public/cft68 ---
CF000 - CFT VERSION -   2.68
CF001 - COMPILE TIME =      0.0083 SECONDS
CF002 -          6 LINES,          6 STATEMENTS
+ ldr f.o
+ a.out
libf @END called
+ rm f.f f.o a.out
+ jar -chsf
COMMAND REPORT
===========

COMMAND      STARTED     USER-CPU     SYS-CPU      I/O-WAIT     ELAPSED      SBU'S
NAME         AT          [SECONDS]    [SECONDS]    [SECONDS]    [SECONDS]
======       =====       ======       ======       ======       ======       ======
jad          14:39:26         0.01         0.03         0.00         0.33         0.00
cat          14:39:26         0.01         0.02         0.00         2.47         0.00
cft68        14:39:31         0.01         0.02         0.00         0.27         0.00
sh           14:39:29         0.01         0.01         0.00         2.34         0.00
sh           14:39:29         0.01         0.01         0.00         2.35         0.00
ld           14:39:32         0.57         0.19         0.00         4.87         0.00
sh           14:39:32         0.01         0.01         0.00         5.20         0.00
a.out        14:39:37         0.00         0.01         0.00         0.42         0.00
rm           14:39:38         0.00         0.02         0.00         0.49         0.00

PROCESS FLOW CHART
=============

parent     ->  child          . . .
=================

jad
cat
sh         ->  sh            ->  cft68
sh         ->  ld
a.out
rm

JOB ACCOUNTING REPORT
===============
Operating System              : Mendota Mendota sysV bak CRAY-2
Job Name                      : test
User                          : cda (1019)
Group                         : cray2 (101)
Report starts                 : 03/24/86 14:39:26
Report ends                   : 03/24/86 14:39:38
CPU Time (User)               :           0.6329 Seconds
CPU Time (System)             :           0.3193 Seconds
I/O Wait Time                 :           0.0000 Seconds
Elapsed Time                  :          12      Seconds
CPU Time Memory Integral      :           0.0310 MWords * Seconds
I/O W-Time Memeory Integral   :           0.0000 MWords * Seconds
Data Transferred              :           2.0087 Mbytes
Phys. I/O Requests            :          36
No. of Commands               :           9
Billing Units                 :           0.0000
```

Figure 6. Sample UNICOS Batch Job Output

```
qstat

batch@navier; type=BATCH; [ENABLED, RUNNING]; pri=16
  0 exit;      1 run;      0 stage;      0 queued;      0 wait;      0 hold;      0 arrive;
            REQUEST NAME        REQUEST ID          USER          PRI     STATE       PGRP
<2 Requests>
      3:            nqscheck        157.navier            rowley          63   RUNNING       241
```

```
qstat -l

batch@navier; type=BATCH; [ENABLED, RUNNING]; pri=16
  0 exit;      1 run;      0 stage;      0 queued;      0 wait;      0 hold;      0 arrive;

   Request      3:  Name=nqscheck    Id=157.navier
       Owner=rowley  Priority=63    RUNNING Process-group=241
   Created at Fri Apr 11 14:48:19 PST 1986
   Mail = [NONE]
   Mail address = rowley@navier
   Owner user name at originating machine = rowley
   Per-proc. permanent file size limit= [2 gigabytes, 2 gigabytes] <DEFAULT>
   Per-proc. memory size limit= [2 gigabytes, 2 gigabytes] <DEFAULT>
   Per-req. memory size limit= [2 gigabytes, 2 gigabytes] <DEFAULT>
   Per-proc. execution nice priority = 4 <DEFAULT>
   Per-proc. CPU time limit= [72000.0, 72000.0] <DEFAULT>
   Per-req. CPU time limit= [360000.0, 360000.0] <DEFAULT>
   Standard-error access mode = SPOOL
   Standard-error name = navier:/u/nas/npo/rowley/nqs/nqschec.e157
   Standard-output access mode = SPOOL
   Standard-output name = navier:/u/nas/npo/rowley/nqs/nqschec.o157
   Shell = DEFAULT
   Umask = 22
```

Figure 7. Qstat Examples

Command	Function
Abort Queue	Signal all requests in a queue.
Create Batch-queue	
Create Pipe-queue	
Delete Queue	
Delete Request	
Disable Queue	Prevent any more requests from being placed in this queue.
Enable Queue	Queue is enabled to accept new requests.
Help	
Purge Queue	All queued requests are purged (dropped) from the queue and are irretrievably lost.
Set Nice-value	Set a best nice-value for a batch queue.
Set Per-Request Cpu-limit	Set a per-request maximum CPU time limit for a batch queue.
Set Per-Request Memory-limit	Set a per-request maximum memory size limit for a batch queue.
Set Run-limit	Change the run-limit of an NQS batch or pipe queue.
Show Queue	Display the status of the specified queue(s).
Shutdown	Shutdown all moving parts of NQS on the local host.
Start Queue	Queue is started and requests in the queue are eligible for selection.
Stop Queue	Stop initialization of requests from a queue.

Figure 8. Qmgr Commands

Software Tools: C on the Cray

chaired by Mary Zosel

Lawrence Livermore National Laboratory
Livermore, California

C is a mature and widely used language, but it is new to the Cray environment. This session included a talk addressing the general direction of development and standardization of C, a series of short reports on user experience with a variety of preliminary versions of the Portable C, and a description of the features in the first release of a CRI supported C compiler. The talks are summarized following.

The Status and Evolution of C
Larry Rosler, AT&T

The C language was an outgrowth of B and BCPL. It evolved through proliferation to diverse applications, and after the development of the portable C compiler PCC, through proliferation to different machines. The porting and diversity led to a variety of additions to the original C language. Remarkable defacto standardization has arisen from conformance of new implementations to the language and environment defined in the book The C Programming Language by Kernighan and Ritchie (often referred to as K&R). Formal standardization efforts are now in progress. Meanwhile development in programming language design has led toward many useful newer language concepts. At AT&T, the design of the C++ preprocessor has addressed addition of new features to the C language.

Standardization grew out of a /usr/group effort. An ANSI committee, X3J11, has been working since 1983 and expects to release a proposed standard for public comment later this year. The language standardized has intentionally made it a goal not to invalidate C programs. Some features which "surprise" new C programmers remain because they are considered part of the language. The committee is adding a number of proven features such as enumeration types and templates of expected function parameter types to aid separate compilation type checking. The definition of templates illustrates the care taken to allow interface to older C programs which lack template information.

At the same time as the standardization effort, a "new C" was developing at Bell Labs. There was debate about a name for the language and, in the spirit of C, they decided the way to get the next name was C++. This language has been described in the book The C++ Programming Language by Stroustrup . It adds many new language features for data abstraction, type checking, and object-oriented programming, such as classes and overloaded operators. C++ is a preprocessor which turns programs written in the extended language to C programs. (Before C++ can be used on Crays, users will have to wait for compiler and system support to remove the restriction on 8 character external names. This is in progress.)

Experience at User Sites
Martin Fouts, NASA Ames
Rich Schultz, IDA
Robin O'Neill, MFECC
Kelly O'Hair, LLNL

A variety of unsupported versions of the Portable C compiler, preprocessor, and library have found their ways to Cray machines. Martin Fouts, NASA Ames, reported that NASA has been using a version of the compiler on the Cray 2. They have found a reasonable correspondence of the compiler to the K&R standard, however they did find that the implementation dependent features were often "backwards" from what they expected. Some of the implementation decisions, especially for character pointers have resulted in the requirement that the user treat C like a strongly typed language, but without any of the usual compiler diagnostics for type violation. The lack of flex (long) names and a debugger has also been a problem to them.

Rich Schultz, IDA, reported that they have been porting Berkeley code for many tools such as LEX, YACC, TCP/IP, etc. to their machine. They have run into two classes of problems. Some problems stemmed from differences between a Cray and a Vax, such as, dependence on 32 bits for integers and the assumption that integer pointers and character pointers have the same format. They also found malloc using the low bit of a word as a flag. Other problems stemmed from implementation limitations, such as the "silent equivalence" of flex names, case insensitivity, loader problems with separately compiled global names, structure packing, and nonstandard calling sequences.

Robin O'Neill, MFECC, has been using the Protable C compiler to port a Unix™ system that sits on top of CTSS. He reported that basically the code was correct, but not particularly efficient. They were finding many problems with the library, such as representation of end-of-file marks, lack of binary file types, and compressed blanks. They were also bothered by linkage incompatibility, lack of a symbol table for debugging, and an incorrect load map.

Kelly O'Hair, LLNL, reported they are using a modified version of the Portable C to write a new ANSI C compiler. This hybrid compiler will use the backend of the existing LLNL CIVIC compiler for language independent code optimization and generation of the existing compiler. The development work is being done in C, on a VAX, and then bootstrapped to the Cray with the Portable C. A number of problems with the Portable C, similar to those already reported, has been found and documented. This list of the problems encountered is available on request. The new compiler is nearing initial release. It can compile a C library and a new preprocessor is complete. The compiler produces complete listings, debug symbol tables, and is expected to vectorize simple loops. It implements most of the features of the proposed ANSI standard. A initial beta release of the compiler is planned for the end of May, with the official "first" version sometime later in the summer.

CRI Support and Plans for C
Tom MacDonald, CRI

The first release of the Cray C compiler (1.0) went out with COS 1.15 and UNICOS 1.0. The Cray C compiler is based on AT&T's Portable C compiler and runs on Cray-1 and Cray X-MP machines. Several changes to the compiler were necessary before it could be released as a Cray supported product. The first change made was to support a calling sequence compatible with the newly developed stack-based calling sequence developed for COS 1.13. Support was added for extended memory addressing on large memory machines. A source listing option was added to the preprocessor that runs under COS and all error messages were documented and added to the C reference manual. The flowtrace option is now fully supported. The memory manager that came with the original C compiler would not work properly with the COS heap and stack manager. Calls to memory manager functions were replaced with calls to equivalent heap and stack manager functions.

The types 'int', 'unsigned', and 'unsigned long' needed to be changed. The type 'int' was a 24 bit entity in the original AT&T compiler. UNICOS developers were porting codes that depended upon having at least 32 bits of precision. The compiler was modified to support 64 bit signed and unsigned integers. The format of the character pointer was changed in order to get the word address right-justified. The byte offset is left-justified which means all pointers in memory have the same internal format.

Many performance issues were also addressed in the first release of the C compiler. The switch statement and long constant generation were the first optimizations added because they were in very localized places in the compiler. Further optimizations required a two pass compiler. The two pass compiler allowed us to perform funtion at a time optimizations such as jump to jump eliminations, and B and T register usage for "auto" variables. Next, the compiler was changed to internally have an infinite number of pseudo registers instead of a few fixed hard registers. This allows for future optimizations such as common subexpression elimination, instruction scheduling, and automatic vectorization of some loops.

Several options that are being studied are C and multitasking, an ANSI standard C compiler, flexnames (long variable names), and symbolic debugging tools.

FPV - a Floating-Point Validation Package

by Jeremy Du Croz, Numerical Algorithms Group Ltd.

1. Motivation

FPV is a software package for validating an implementation of floating-point arithmetic. By 'validation' we mean simply an experimental verification that floating-point arithmetic has been correctly implemented according to its specification. FPV must be supplied with the essential parameters of the specification and attempts to verify that the arithmetic conforms to these parameters by probing for errors as best we know how. Different implementations of arithmetic may pass the tests according to more or less stringent criteria: the 'best' implementations (such as those that conform to the IEEE standard) will satisfy the most stringent criteria.

Floating-point arithmetic is the basis of almost all scientific and engineering computation. Undetected errors in floating-point arithmetic can lead to unexpected program-failures, many hours wasted on the investigation of puzzling results, and worst of all, to incorrect results being accepted as valid.

Here are some examples of errors that have been detected in production systems (most have now been corrected).

Example 1 (Interdata/32):

```
        SMALL =  0.5**256

             . . .

        X  =  1.0

             . . .

        IF (X.LE.SMALL) GO TO 900
```

The branch to 900 was taken!

Example 2 (ICL 2988):

```
        X  =  5.0 E-78

             . . .

        Z  =  X*X
```

The computed value of Z was 0.335...!

Example 3 (CDC 7600, Cray-1, CDC Cyber 205):

Occasional results in double precision have been no more accurate than single precision: this undermines the reliability of using double precision as a check on the accuracy of single precision.

Who needs to test floating-point arithmetic?

a.) Manufacturers of floating-point hardware:

* to test simulations of hardware designs
* to test prototypes of floating-point chips or boards
* to check the integrity and reliability of each production model

b.) Developers of floating-point software (including microcode):

* to debug their code
* for quality-assurance after each subsequent revision

c.) Managers of scientific computing installations:

* for benchmarking the accuracy of systems under evaluation
* as part of an acceptance test for newly acquired systems
* as a daily check against intermittent faults in the hardware
* to check that hardware and software upgrades have been correctly installed

d.) Implementors of scientific software (e.g. NAG)

* to check the arithmetic characteristics of the systems on which their software is mounted
* to ensure the reliability of those systems

NAG views FPV as the first of a suite of tools for checking various aspects of the computing environment of any machine, to ensure that it can support the implementation of NAG Scientific software.

2. Previous Work

Schryer (1981) developed a program FPTST (about 12,000 lines of Fortran) which tests specifically for conformance with a model of floating-point arithmetic developed by W.S. Brown (1981). Coonen (1984) developed test programs and data designed specifically to test implementations of the IEEE standard. Paranoia, by W. Kahan, (see Karpinski, 1985) is a program which sets out to diagnose, and pass judgement on, the arithmetic properties of a machine, about which nothing need be known in advance.

All the above adopt a highly selective approach to testing, as does FPV; however, with care and good design, this is much more effective than testing millions of operations chosen at random. Only empirical testing is considered here: it offers no absolute guarantee of correctness, but in practice can give a very high degree of assurance.

3. Design of FPV

FPV has been developed at NAG Central Office in collaboration with Dr. B.A. Wichmann of the National Physical Laboratory, under a contract funded by NAG Ltd., and the Department of Trade and Industry. It is based on many of the ideas used by Schryer in FPTST (with advice and encouragement from Schryer and with permission of A.T.& T. Bell Laboratories).

In particular FPV follows Schryer in its strategy for selecting numbers to use as test operands. On a binary machine Schryer uses numbers whose mantissae (or fractions or coefficients) have the following patterns:

$$\begin{array}{c} .100\ldots00100\ldots000 \\ .111\ldots11100\ldots000 \\ \uparrow \\ i \end{array}$$

where the position of the i-th bit can range from 1 up to P (the number of bits in the mantissa). FPV can use other patterns of mantissae, but they have hardly ever revealed any properties of the arithmetic which have not been revealed by the two patterns above.

FPV can test the following basic arithmetic operators:

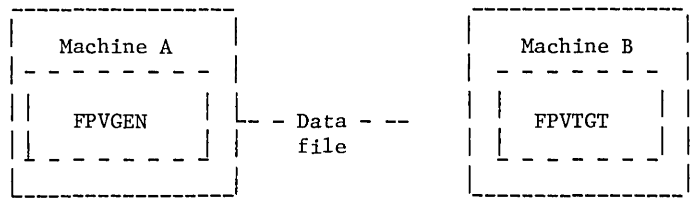
addition and subtraction	x +or- y
multiplication	x*y
division	x/y
square root	sqrt(x)
negation	-x
absolute value	$\|x\|$
comparisons	x=y, x.ne.y, x<y, x>y, x<=y, x>=y

Both scalar and vector operations can be tested.

FPV allows the floating-point number system to have arbitrary base, precision and exponent range. The presumed values for these parameters must be supplied to FPV which attempts to verify experimentally that they are correct. If a few system-specific modifications are made to the programs, FPV can test whether the overflow and underflow flags are set correctly. FPV does not test integer arithmetic, conversions between integer and floating-point formats, or floating-point operations in mixed precision.

FPV can test whether the results are exactly correct according to one of a choice of commonly used rounding rules - including all those specified in the IEEE standard (IEEE, 1985). Alternatively, if the rounding rule is unknown or not one of those provided, FPV tests whether the results lie within the narrow bounds defined in the model of floating-point arithmetic developed by W.S. Brown (the 'Brown model').

In order to facilitate testing in as wide a variety of environments as possible, FPV allows the testing procedure to be split into two phases. In '<u>two-phase</u>' mode, one program FPVGEN generates a file of test data; a second program FPVTGT, usually running on a different machine, reads the file and performs the tests.



The program FPVTGT is comparatively short and simple, and is not difficult to adapt to different environments (even if this involves translating it into a different language). FPV can also operate in '<u>all-in-one</u>' mode, in which the program FPVGEN does not write a file, but immediately performs the test itself, all on one machine.

The programs FPVGEN and FPVTGT are currently written in both standard Fortran 77 and ISO standard Pascal, level 1. They therefore require a suitable compiler to be available and they test the arithmetic as 'seen' through those languages. A few machine-specific modifications may be needed to make the programs completely robust. In order to test arithmetic on machines which do not have a Fortran or Pascal compiler, or to test arithmetic as seen through a different language (e.g. Basic, Ada), it is necessary to translate all or part of the program FPVTGT into a suitable language.

4. <u>Experience with FPV</u>

Errors that have already been detected by FPV include:

1. DEC VAX-11/750: error in division under VMS 3.7 in the emulation software for arithmetic on G-floating and H-floating numbers.

2. ICL 2988: see Example 2 in Section 1.

3. IBM PC with 8087 co-processor and Microsoft 3.2 compiler: last-bit errors in emulation software for square-root.

4. Cray-1S: the arithmetic had been tested by Schryer and was believed to be free of errors but the routines for double precision arithmetic were rewritten at COS 1.14: errors appeared in double precision vector-by-vector multiplication. Errors have also been detected in the single precision vector square-root routine and in the triple precision addition routine.

5. CDC 7600: error in double-precision comparison with the FTN 5 compiler (this had been detected two years earlier by Schryer's program but not yet corrected).

6. CDC Cyber 205: error in double precision comparison (essentially the same as on the CDC 7600); also unnormalized results produced from double precision multiplication, division and negation.

In addition to detecting gross errors in the arithmetic, FPV has also proved useful in determining the precise behaviour of the arithmetic on machines for which the manufacturers' documentation is inadequate. This second use is just as important as the first: while gross errors can be corrected, the need to know the precise properties of the arithmetic will remain. FPV is a tool to help achieve both greater reliability and more accurate information.

References

Brown W.S. (1981). A simple but realistic model of floating-point computation. ACM Trans. Math. Software 7, 445-480.

Coonen J.T. (1984). A compact test suite for P754 arithmetic - Version 2.0. Chapter 10 of Ph.D. Thesis, University of California, Berkeley.

IEEE (1985). Standard for Binary Floating-point Arithmetic. ANSI/IEEE Std 754-1985.

Karpinski R. (1985). Paranoia: a floating-point benchmark, Byte 10, no. 2, 223-235.

Schryer N.L. (1981). A test of a computer's floating-point unit. Computer Science Technical Report No. 89. A.T. & T. Bell Laboratories, Murray Hill, NJ.

Availability

Further information about the availability of FPV can be obtained from:

Numerical Algorithms Group Inc.
1101 31st Street, Suite 100
Downers Grove, IL 60515-1263;

or, if outside N. America, from:

Numerical Algorithms Group Ltd.,
256 Banbury Road
Oxford OX2 7DE
England

COMPARISON OF NAG, IMSL AND CRAY LIBRARIES

Chris Lazou & David Winstanley

University of London Computer Centre
England

The talk consisted of a comparison of IMSL 9.2, NAG11F, Cray Applications and SCILIB. A series of PIE charts were presented showing the composition of each library when classified under the headings of Arithmetic, other mathematical, statistics and general.

SCILIB

The SCILIB library does not have any statistical routines, the ones illustrated are in fact the Random Number generator routines in ARLIB. The strong Arithmetic content derives from the BLAS routines and LINPACK, the mathematical content comes from EISPACK and LINPACK and the general content is from the Sorting and Searching routines.

Cray Applications

The Cray Applications Library has a predominantly mathematical content with routines from FITPACK, FFTPACK, EISPACK and LINPACK.

Note: Routines in these Public Domain software libraries have been specially tailored for the Cray and are in general CAL coded.

Some of the EISPACK and LINPACK routines are in both the Cray Applications Library and SCILIB.

IMSL

IMSL has a strong statistical bias and is moderately well endowed in all the other areas.

NAG

The mathematical content is dominant here and the Arithmetic and General subject areas are of equivalent strengths to IMSL.

Combination

We should point out that both the NAG and IMSL libraries have approximately equal number of user callable routines:

NAG with 524
and IMSL with 517

whereas the Cray Applications (Maths and Stats libraries) and SCILIB have 276 and 239 routines respectively.

Bar Charts

The bar charts illustrated give a breakdown of the number of routines from each of the libraries within their subject heading.

Arithmetic

Figure 1 shows that IMSL is dominant in the matrix arithmetic section. This is due to the duplication of routines for the different storage modes.

NAG has more decomposition routines than IMSL with unique routines for the Hessenberg form.

NAG has unique headings for complex arithmetic (with square root, modulus and quotient routines) and also for orthogonalisation with its GRAM-SCHMIDT routine.

IMSL has routines for scalar arithmetic, which include the decomposition of an integer into its prime factors and some extended precision routines.

Only SCILIB and IMSL have a formal set of BLAS routines, which includes the index of the maximum absolute value of a vector, a vector multiplied by a scalar (known as a SAXPY), the Euclidean norm of a vector, the DOT product, and the scaling and swapping of vector elements. The BLAS also includes routines for Givens Rotations. NAG, however, have defined a set of Kernel routines which are documented in an article by Jeremy Du Croz in issue 2 of the 1983 NAG Newsletter.

Mathematics

Figure 2 shows that NAG is particularly strong in its Ordinary Differential Equations Chapter and its Optimisation Chapter. It has unique headings for Integral Equations and Summation of Series (excluding Fourier Transforms which are included under Filters and Transforms).

IMSL has no particularly strong subject here, however it does have routines for nearly all the subjects within this area.

The Cray libraries are strong for Eigen-System Analysis, Filters and Transforms and Interpolation and Approximation with EISPACK, FFTPACK and FITPACK routines. The Cray Applications library also includes FISHPACK which is a set of routines for evaluating Elliptical P.D.E.s.

Statistics

Figure 3 shows that IMSL is dominating with unique headings for Categorical Data Analysis, Multivariate Statistics and Sample Surveys. The Regression routines also include unique routines for L1 and norm Non-Linear regression. Both NAG and IMSL will estimate Box-Jenkins Times models as well as Frequency Domain models.

NAG has some unique routines in Operational Research for Quadratic Programming and Transportation problems.

IMSL has a full range of Statistical distributions.

General

Figure 4 shows that under special functions IMSL has unique routines for Kelvin functions, whereas NAG will evaluate Fresnel integrals.

IMSL has a number of useful utility routines for printing matrices of various storage modes. Both IMSL and NAG will produce line printer plots, however NAG has a full Graphical Supplement available on the Cray-1S with a GINO-F interface.

When library routines are used in large scale production runs, users are advised to check their timings where duplicate routines are available, since both improved performance and significant cost reductions can be achieved.

Examples of how to achieve relative timings by using CFT's Flow-trace, with library routines being called from dummy user routines, were given and a few results are listed in the following four examples.

Example 1:

Subject: Multiple Linear Regression

Library:	Cray Appl.	NAG/OF	IMSL 9.2
Routine name:	RGRSN1	G02CJF	BECOVM & RLMUL
For size:	(P=4, n=21)		
CPU time:	0.000717	0.002581	0.001299
Time ratio:	1	3.5	1.8
For size:	(P=4, n=150)		
CPU time:	0.001926	0.014387	0.002130
Time ratio:	1	7.5	1.1
For size:	(P=6, n=150)		
CPU time:	0.002436	0.018551	0.003138
Time ratio:	1	7.6	1.3

Example 2:

Subject: Matrix Multiplication

Library:	Cray Appl.	NAG10F	IMSL 9.2
Routine:	MXM	F01CKF	VMULFF
For size:	[20:30] [30:20]		
CPU time:	0.004114	0.018858	0.065378
Time ratio:	1	4.6	15.9
For size:	[200:300] [300:200]		
CPU time:	0.167643	0.364754	1.483208
Time ratio:	1	2.2	8.8

Example 3:

Subject: Quadrature

Library:	Cray Appl.	NAG10F	IMSL 9.2
Routine name:	ADQUAD	D01AHF	DCADRE
CPU time:	0.000251	0.000372	0.000307
Time ratio:	1	1.5	1.2

Example 4:

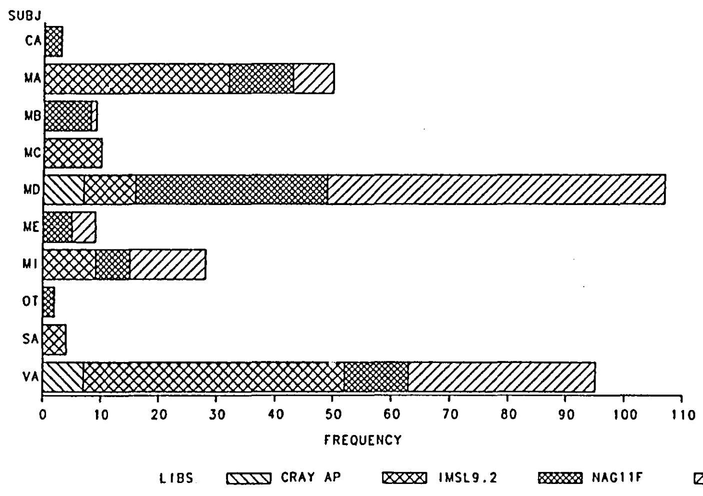
Subject: Sort vector of 1000 real numbers

Library:	Cray Appl.	NAG10F	IMSL 9.2
Routine name:	-	M01AAF	VSRTA
CPU time:	-	0.012941	0.007592
Time ratio:	-	1.7	1

Figure 1

Comparison of NAG11F, IMSL9.2 and CRAY M&S libraries

APPLIC=A

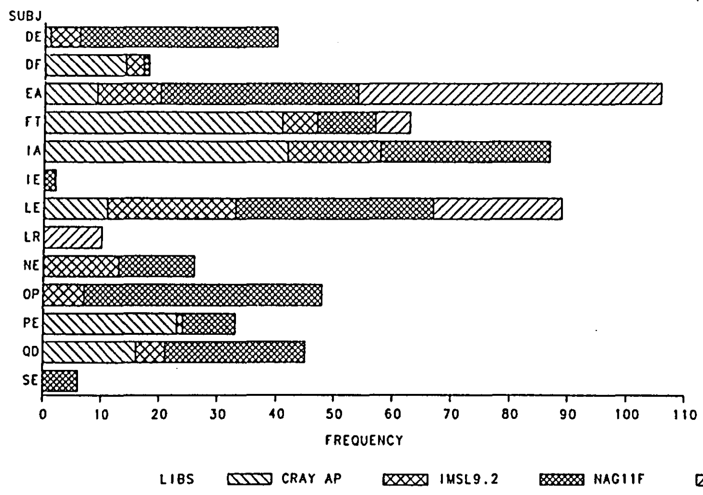


SUBJ	FREQ	CUM. FREQ	PERCENT	CUM. PERCENT
CA	3	3	0.95	0.946
MA	50	53	15.77	16.719
MB	9	62	2.84	19.558
MC	10	72	3.15	22.713
MD	107	179	33.75	56.467
ME	9	188	2.84	59.306
MI	28	216	8.83	68.139
OT	2	218	0.63	68.770
SA	4	222	1.26	70.032
VA	95	317	29.97	100.000

Figure 2

Comparison of NAG11F, IMSL9.2 and CRAY M&S libraries

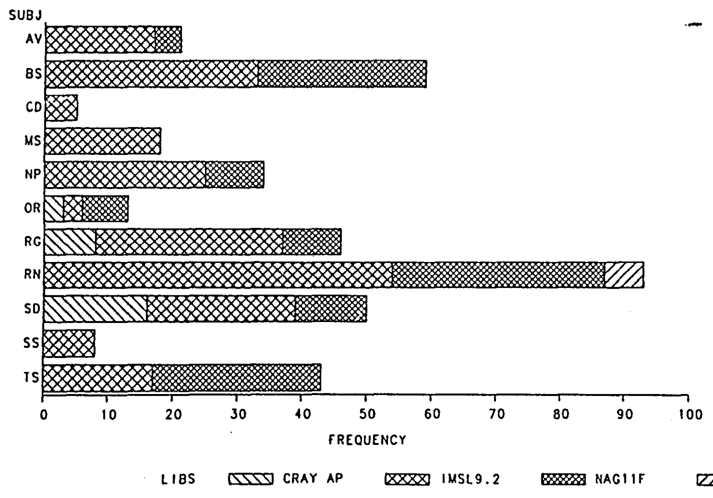
APPLIC=M



SUBJ	FREQ	CUM. FREQ	PERCENT	CUM. PERCENT
DE	40	40	6.981	6.981
DF	18	58	3.141	10.122
EA	106	164	18.499	28.621
FT	63	227	10.995	39.616
IA	87	314	15.183	54.799
IE	2	316	0.349	55.148
LE	89	405	15.532	70.681
LR	10	415	1.745	72.426
NE	26	441	4.538	76.963
OP	48	489	8.377	85.340
PE	33	522	5.759	91.099
QD	45	567	7.853	98.953
SE	6	573	1.047	100.000

Figure 3

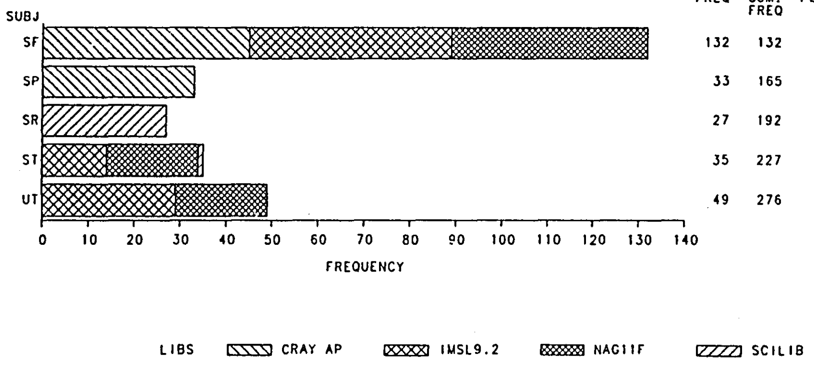
Comparison of NAG11F, IMSL9.2 and CRAY M&S libraries
APPLIC=S



SUBJ	FREQ	CUM. FREQ	PERCENT	CUM. PERCENT
AV	21	21	5.385	5.385
BS	59	80	15.128	20.513
CD	5	85	1.282	21.795
MS	18	103	4.615	26.410
NP	34	137	8.718	35.128
OR	13	150	3.333	38.462
RG	46	196	11.795	50.256
RN	93	289	23.846	74.103
SD	50	339	12.821	86.923
SS	8	347	2.051	88.974
TS	43	390	11.026	100.000

Figure 4

Comparison of NAG11F, IMSL9.2 and CRAY M&S libraries
APPLIC=G



SUBJ	FREQ	CUM. FREQ	PERCENT	CUM. PERCENT
SF	132	132	47.83	47.83
SP	33	165	11.96	59.78
SR	27	192	9.78	69.57
ST	35	227	12.68	82.25
UT	49	276	17.75	100.00

STATIC DEBUGGING OF MULTITASKING PROGRAMS

Bill Appelbe

Electrical Engineering and Computer Sciences Dept. C-014
University of California, San Diego
La Jolla, CA 92093

Charlie McDowell

Computer and Information Sciences Dept.
University of California, Santa Cruz
Santa Cruz, CA 95064

ABSTRACT

The use of asynchronous multi-tasking constructs in high-level languages to increase performance has led to a new class of program 'bugs', including race conditions when two tasks access/update shared variables. Such bugs are notoriously hard to detect because they are non-deterministic, and can be obscured or undetectable by conventional run-time dumps and traces.

We have developed a static analyzer for multi-tasking Fortran, which finds potential bugs by simulating the execution of concurrent tasks. The analyzer also provides information on the reachable states of a concurrent program, to assist in optimization and algorithm analysis.

This talk will discuss the need for static debugging, and describe the design and use of the debugger.

1. Introduction

The use of asynchronous multi-tasking constructs in high level languages to increase the amount of exploitable parallelism has created a new class of program errors. These errors are particularly difficult to find using conventional debugging techniques due to their frequently non-deterministic behaviour. A program may give one result for one execution and give a different result on a subsequent execution using the same input data. These variations are due to uncontrollable timing interactions with other parts of the computer system. Even when the error can be reliably reproduced, when diagnostic code is inserted the error may change in appearance or disappear entirely.

Data flow analysis techniques have been developed for conventional (synchronous) programs, to detect certain types of errors and for performing program optimization.[1,2] The primary errors detectable by these techniques are references to undefined variables, and definitions of variables that are never referenced. Some preliminary work has been done in the area of extending these techniques to parallel programs.[3,4,5]

This work was supported by DOE contract number W-7405-Eng-36 through Los Alamos National Laboratory.

[1] L. J. Osterweil and L. D. Fosdick, "DAVE - A Validation, Error Detection, and Documentation System for Fortran Programs," *Software-Practice and Experience*, vol. 6, pp. 473-486, 1976.

[2] A. V. Aho and J. D. Ullman, *Principles of Compiler Design*, Addison-Wesley, 1977.

[3] R. N. Taylor and L. J. Osterweil, "Anomaly Detection in Concurrent Software by Static Data Flow Analysis," *IEEE Trans. on Software Eng.*, pp. 265-278, May 1980.

[4] G. Bristow, C. Drey, B. Edwards, and W. Riddle, "Anomaly Detection in Concurrent Programs," *Proc. 4th Int. Conf. Software Eng.*, 1979.

A concurrent system consists of a collection of tasks and shared data. In analyzing a concurrent system our goal is to determine all statements which could potentially execute in parallel. When combined with standard dataflow analysis, this will allow for the detection of the following additional *anomalies*, or potential errors:

synchronization
: These are errors that occur in the synchronization of task (or parts of task) execution, such as deadlock.

parallel access
: Two simultaneous accesses (other than two reads) to a piece of shared data by concurrently executing processes. The result of such a parallel access is non-deterministic and in most situations is an error.

Taylor and Osterweil have published a set of algorithms[6] that operate on a flow graph that has been augmented to handle parallel processes. These algorithms are very similar to conventional data flow analysis algorithms. They claim that all of the above anomalies can be detected when the augmented flow graph is suitably restricted. Their algorithm has several important restrictions:

- "a process may not be scheduled to execute in parallel with itself". This precludes directly implementing their algorithm for our target language (HEP Fortran), which allows multiple copies of a task to be created at run-time.
- A second restriction arises from the use of a "process augmented flowgraph" in which the flowgraphs for each task are connected with special edges to indicate synchronization constraints. It is correctly stated in the paper that "it is impossible to create a fixed static procedure capable of constructing the PAF of any program written in a language which allows run-time determination of tasks to be scheduled and waited for." In our algorithm a limited form of run-time task scheduling is supported.

An algorithm for the detection of parallel paths is fundamental to any analysis of anomalies resulting from execution of parallel paths. The Taylor and Osterweil results assume the existence of such an algorithm. In a more recent paper,[7] Taylor goes on to describe such an algorithm that is being implemented for a subset of Ada.[8] This recent algorithm by Taylor, served as the starting point for our work. The primary addition is the notion of *families* and *clans* of tasks used to handle multiple (possibly arbitrarily many) copies of a single task flowgraph executing concurrently.

Section 2 of this paper briefly describes the concurrency primitives that are supported by the analysis algorithm. Sections 3, 4 and 5 describe the analysis algorithm.

2. Concurrency Primitives

The ideal characteristics of a set of concurrency primitives to support parallel algorithms are:

Completeness
: The primitives should be sufficient for a wide range of parallel algorithms and environments. The primary environment in which the primitives will be used is scientific programming. Parallelism may occur at various levels, however the primitives are designed for task-level parallelism with shared memory. Thus, there are no explicit operations for parallel or pipelined arithmetic units, and data can be shared between tasks.

Efficiency
: At compile-time - the primitives should be easily translated into code which executes efficiently on a wide range of parallel architectures.

[5] R. N. Taylor, "A General-Purpose Algorithm for Analyzing Concurrent Programs," *CACM*, vol. 26, no. 5, pp. 362-376, May 1983.

[6] R. N. Taylor and L. J. Osterweil, "Anomaly Detection in Concurrent Software by Static Data Flow Analysis," *IEEE Trans. on Software Eng.*, pp. 265-278, May 1980.

[7] R. N. Taylor, "A General-Purpose Algorithm for Analyzing Concurrent Programs," *CACM*, vol. 26, no. 5, pp. 362-376, May 1983.

[8] R. N. Taylor, *personal communication*, 1984.

Simplicity

The primitives should be easily understood, learnt, and used by programmers. This implies that the primitives should have a simple syntax and semantics, and the primitives should not present users with unnecessary options. The primitives should also be comparatively easy to treat analytically, so that the parallel behaviour of algorithms can be determined automatically.

Obviously there is a conflict between such goals as simplicity and completeness, so that any set of extensions must necessarily be a compromise. The following is a summary of the primitives supported in the current algorithm and it's implementation at UCSD. The primitives are converted to the target language (HEP or CRAY Fortran) by a preprocessor. A complete discussion of the primitives can be found in another report by the authors.[9] Similar sets of primitives have been reported elsewhere.[10, 11]

3. Concurrency Analysis

For our analysis two classes of global data will be considered:

Atomic Data

Atomic Data, which is used for task synchronization, consists of data types such as events, counters, and locks, which can be accessed only by means of indivisible *concurrent operations* such as wait_event or increment_counter.

Non-Atomic Data

Non-Atomic Data, which is used for task communication, consists of data which can potentially be concurrently read and written, such as a shared message pool.

The analysis is based upon the program *control flowgraph*, in which each node represents a sequence of 'straight-line' code terminated by a transfer of control, or a concurrent operation. Arcs in the control flowgraph represent sequential and branch transfers of control, and task creation. The concurrent behaviour of a *concurrent system CS* is defined by a *synchronization graph SG*. The synchronization graph is a compressed version of the control flowgraph. All nodes in the control flowgraph can be classified as either sequential or concurrent operations. A *sequential path* in the control flowgraph is a path in which all intermediate nodes are sequential. The synchronization graph is the subgraph of the control flowgraph consisting of all concurrent nodes, and arcs representing all the sequential paths between these nodes. Each node in *SG* corresponds to an operation upon a shared atomic variable, or to task creation and termination. Each arc in *SG* is labelled by operations upon local and shared non-atomic data.

Our goal is to determine the *concurrency history graph CHG*, whose nodes are *concurrent states*, and whose arcs are *history transitions*. Each node in *CHG* describes one reachable concurrent state of the system. A concurrency state *S* consists of two components:

- a *control-state TS*, and a
- a *data-state DS*.

The control state consists of a dynamic list of *task familys*. A task family *Ti* consists of a set of 'identical' tasks. Each task has the same task type *TiT* and current state *TiS*. Each task also has a unique task id *TiI*, which is assigned when the task is created[1]. The task count, *TiC*, denotes the number of tasks of this type currently in the same state.

Every task has an entry point, at which it commences execution. The task type represents the synchronization graph entry point of the task family. Two tasks have the same type only if they have the same subgraph. The task state represents the execution state, i.e., the current node in the synchronization graph

[9] W. F. Appelbe and C. E. McDowell, "Language Primitives for Parallel Numerical Algorithms," *unpublished.*

[10] E. L. Lusk and R. A. Overbeek, "An Approach to Programming Multiprocessing Algorithms on the Denelcor HEP," *ANL-83-96, Argonne National Laboratory.*

[11] H. F. Jordan, "Using the Force to Write Parallel HEP Programs," *unpublished,* June 1984.

1 Task id's are not normally considered as components of the task state, since two concurrency states with differing assignments of task id's are equivalent. Task id's are assigned when a task is created, and may be used to distinguish between tasks (e.g., when waiting for task termination).

executed by the task. Since nodes represent atomic concurrent operations, the task state may either be before or after completing the operation. Each concurrent operation consists of two states, referred to as *pre-* and *post-* transition states (to emphasize the distinction, pre-transition states are denoted by $\overline{state\_name}$ and post-transition states are denoted by *state_name*). At a pre-transition state a task is waiting to access the global state. At a post-transition state a task has completed all accessing and updating of the global state for the concurrent operation. Thus, the concurrent operation actually occurs during the transition from a pre- to a post-transition state. Transitions from pre- to post-transition states are referred to as *atomic transitions,* whereas transitions from post- to pre-transition states are referred to as *non-atomic transitions.* A task thus executes an interleaved sequence of atomic and non-atomic transitions.

The entry point of any task is a post-transition state, as task creation is considered an atomic operation. Also, task termination is also an atomic operation, but the post-transition state consists of removing the task from the control state *TS*

The data state *DS* consists of a set of shared atomic variables (locks, events, barriers, and counters). Thus, any concurrent state can be represented as follows:

$$S=(TS,DS)\ where$$

$$TS=[Ti]\ where\ Ti=(TiT,TiS,TiC)$$

$$DS=([L\_i:i=1..LMAX],[E\_i:i=1..EMAX],$$
$$[B\_i:i=1..BMAX],[C\_i:i=1..CMAX])$$

The successors a concurrency history node are thus determined from the state of each Ti[2] as follows:

Atomic Transition

If *TiS_* is a pre-transition state, and the transition can be performed (based upon the data state *DS*), generate a new state, by updating *TiS_* to the post-transition state *TiS*. The update of *DS*, and the number of tasks in the family which advance to the new state depends on the semantics of the concurrent operations as defined below.

Non-Atomic Transition

If *TiS* is a post-transition state, generate new states, by updating *TiS* to all successors states (each of which is a pre-transition state). If more than one task is at that state the rule for generating the successor states is:[3]

$$(TiT,TiS,TiC)\rightarrow[(TiT,TiS,TiC_0),\cdots$$
$$(TiT,TiS\_{1},TiC_1),\cdots$$
$$(TiT,TiS\_{j},TiC_j),\cdots$$
$$(TiT,TiS\_{M},TiC_M)]$$

where

$$succ(TiS)=[TiS\_{1},\cdots,TiS\_{M}]$$

$$\sum_{j=0}^{M}TiC_j=TiC$$

Thus, each task in the family can advance to any one of the M successor states TiS_{j}, or remain in the initial state. Because the number of concurrent successor states increases exponentially with M and TiC (there are

$$\left[\frac{TiC+M-1}{M}\right]$$

2 The *wait_barrier* operation concurrently affects the state of several tasks, as defined below.
3 All tasks cannot advance, otherwise anomalies caused by the transition $TiS\rightarrow TiS\_{j}$ and a later transition by a member of this task family might not be detected.

successors), the exact number of tasks in each successor state are represented symbolically. Tasks $TiS, TiS\_{}_1, \cdots, TiS\_{}_M$, are said to be members of a *task clan*, subject to the *clan size constraint* that

$$\sum_{j=0}^{M} TiC_j = TiC \ \ and \ TiC_j \geq 0$$

Clans may consist of a single task family. Thus, every task is a member of a unique clan. In many cases the number of tasks in a given state may be indeterminate. Such states are referred to as * *states*, and if $TiC = *$, then $TiC_j = *$ for all successors. The basic rule governing the derivation of successor states is that clan size constraints must be preserved, except for explicit task creation and termination operations. Task clans can be represented by a list of task family members, together with the clan size (constant, or unbounded, i.e., ∞). Each member of the clan can either have a fixed task count, or more commonly an indeterminate task count (denoted by *), but in either case these are no greater than the clan size. Task ids are associated with task clans, as any permutation of task ids within a given clan is permitted.

4. State Transitions

The above rules determine how new concurrency states are generated. In addition, it is necessary to determine when two concurrency states are identical. Two states are identical if their data states and control states are identical. Control states are identical if there are the same number of tasks in each state, and if the task clan constraints are identical. Two task families Ti and Tj in the control state TS can be *merged* if

- the task types are the same, i.e., $TiT = TjT$, and
- the task states are the same, i.e., $TiS = TjS$, and
- either both task families are in the same clan, or neither task is in a clan.

In the either case, the task count of the merged task family is simply $TiC + TjC$ (where * + N = * + * = *), and in the former case the task clan constraint is updated by removing the two tasks and inserting the merged task in the clan list.

The principle of task merging applies iteratively to more than two task families. Merging sequences of task families is unnecessary in practice, because the control state is kept in a *canonical form*, by merging families whenever a new state is generated.

The rules given below are for determining Atomic Transitions for each of the concurrent operations currently supported. The rules consist of

- an initial precondition on DS,
- an update to DS, and TS.

Unless otherwise stated, the default rule for task state updates applies as follows:

$$(TiT, TiS\_, TiC) \rightarrow [(TiT, TiS\_, TiC_0), (TiT, TiS, TiC_1)]$$

where $TiC = TiC_0 + TiC_1$.
If $TiC = 1$ then $TiC_1 = 1$ and $TiC_0 = 0$ (i.e., a single task advances to the new state, and hence the old family is deleted).
If Ti was a member of a clan, then both successors are members of the same clan and the clan size constraint is preserved.

4.1. Task Creation

1.
   $TiS = create\_task$ (family size, fs = 1)
   $TiS = create\_task\_family$
   Pre-condition: NTASKS+$TiC * fs$ < MAX_SYSTEM_TASKS
   DS Update: NTASKS $\leftarrow$ NTASKS+$TiC * fs$, and
   $E_{TiI} \leftarrow CLEAR$ for each task id in the family.
   TS Update: TS is updated by default, then
   $TS \leftarrow TS \cup [(Tj\, created\_task\_entry, \text{Tj}\, init\_state, \text{TjC})]$
   $TjC = TiC * fs$

The newly created tasks can be merged into an existing family if that family is not in a clan, otherwise they form a new family. Since the *init_state* is post-transition state, the tasks in this state can be immediately advanced by a non-atomic transition to the successor pre-transition states.

2.

TiS=task_return
Pre-condition: *none*
DS Update: NTASKS ← NTASKS-*TiC* , and E_{TiJ}←*POSTED* for each task id in the family.
TS Update: *TS* is updated by default, except that the successor state *Tj task_return* is a null state and can be deleted from *TS*.

3.

TiS=static_parallel_do
TiS=dynamic_parrallel_do
Pre-condition: NTASKS+*TiC* **loop_tasks* < MAX_SYSTEM_TASKS
DS Update: NTASKS ← NTASKS+*TiC* **loop_tasks* , and
E_{TiJ}←*CLEAR* for each task id in the family.
TS Update: *TS* is updated by default, then
TS←*TS* ∪[(*Tj loop_task_entry*,Tj *init_state*,TjC)]
TjC = *TiC* **loop_tasks*
Static and dynamic parallel do loop tasks are treated as conventional *create_task_familys*, except insofar as anomaly detection and task termination are concerned.
Within the body of a *loop_task*, the *loop_var* will have a distinct value for each task. Thus, concurrently accessing locations $a_1 \times loop\_var + b_1$ and $a_2 \times loop\_var + b_2$ is anomalous only if $(a_2 \times n2) - (a_1 \times n1) = b_2 - b_1$ for some $n1 \neq n2$ in the range of the loop variable.
Static and dynamic parallel loops differ only in the implementation of loop incrementation. The value of the *loop_var* is non-deterministic in both cases, but in the case of a static scheduled loop the difference between *successive* values of the loop index is equal to the number of loop_tasks.
Static and dynamic parallel loop tasks have an implicit barrier upon task return (til all loop tasks have completed).

4.2. Events

1.

TiS=clear_event
Pre-condition: *none*
DS Update: *E_i*←*CLEAR*
TS Update: *default*

2.

TiS=post_event
Pre-condition: *none*
DS Update: *E_i*←*POSTED*
TS Update: *default*

3.

TiS=wait_event
Pre-condition: *E_i*≡*POSTED*
DS Update: *none*
TS Update: *default*
This includes "*wait_task_termination*"

4.3. Common Do Loops

1.

TiS=initialize_common_do
TiS=begin_common_do
TiS=end_common_do

Pre-condition: *none*
DS Update: *none*
TS Update: *default*
A common do has no effect upon the data state, except insofar as anomaly detection is concerned. The effect upon anomaly detection is as described above for pre- and self-scheduled do loops. It is an error to begin a common do prior to initialization (which sets the common do loop bounds).

4.4. Locks

1.
$TiS = clear\_lock$
Pre-condition: *none*
DS Update: $L\_i \leftarrow OFF$
TS Update: *default*

2.
$TiS = lock\_off$
Pre-condition: *none* (though E_i == OFF is erroneous)
DS Update: $E\_i \leftarrow OFF$
TS Update: *default*

3.
$TiS = lock\_on$
Pre-condition: $E\_i \equiv OFF$
DS Update: $E\_i \leftarrow ON$
TS Update: *default*, except that
$TiC_1 \equiv 1$ (i.e., only a single task advances to the new state).

4.5. Barriers

1.
$TiS = initialize\_barrier$
Pre-condition: *none* (though it is erroneous if any task is waiting on the barrier)
DS Update: $B\_i \leftarrow barrier\_count$
TS Update: *default*

2.
$TiS = wait\_barrier$
Pre-condition: *none* (though B_i *uninitialized* is erroneous)
DS Update: $B\_i \leftarrow B\_i - 1$
TS Update: if $B\_i \equiv 0$ then *every* task at the barrier is advanced to a successor state, and the barrier is reset ($B\_i \leftarrow barrier\_count$.

5. Anomaly Detection

The two types of parallel anomalies mentioned in the introduction are synchronization and parallel access. Anomalies of both types can be detected by examining the *concurrency history graph.*

Deadlocks, the main form of synchronization anomaly, correspond to concurrent states with no successor states. Parallel access anomalies, (e.g. concurrent read/writes to the same variable) are detected by examining in turn, each pre-transition concurrent state, or *canonical concurrent state,* in which all task states are pre-transition states (post-transition states are reached when a single process updates the global state atomically, no anomalies occur). Anomalies can exist if there are two or more task transitions which gave rise to the same concurrent state. An anomaly exists if the intersection of all variables written in the task transitions is non-empty, or if the intersection of variables written and read in the task transitions is non-empty.

A third class of anomaly arises if a variable whose value is non-deterministic is used in a situation where a deterministic value is expected. Such anomalies are difficult to detect, as they depend on the semantics of the algorithm. For example, if I was a "common_do" index, then its value in a given loop iteration is non-deterministic, and assignment of I to a local variable, or a conditional statement based on I is probably an error and therefore should be reported as an anomaly.

The determination of anomalies for simple variables is straightforward, since Fortran uses static storage allocation. The determination of anomalies for array references is simplified if the array index is a counter, or a parallel or common do loop index (every such value is unique). Although array reference anomalies are in general undecidable, our technique reports all such undecidable anomalies.

6. Conclusion

A technique for analyzing parallel programs to detect anomalies in concurrent algorithms has been presented. The algorithm of R.N. Taylor has been significantly modified to allow for efficient representation of concurrency histories when greater than one (possibly arbitrarily many) copies of a task may be executing concurrently. This is precisely the situation that arises in most parallel numerical algorithms.

A prototype tool to perform the concurrency analysis has been developed at UCSD in cooperation with Los Alamos Laboratories, and is currently being extended to generate reports of all anomalies detected by the analysis. The tool is targeted to analyze programs written in Fortran-77 extended with the concurrency primitives described. A macroprocessor has been developed to translate the extended fortran into HEP-Fortran for the Denelcor HEP.

Acknowledgements

We would like to thank Olaf Lubeck at Los Alamos for getting us started on this work. Alan Finke and Terri Liebowitz at UCSD have contributed greatly to the current implementation.

Dynamic Debugging for Multitasking

by

James Tabor

Los Alamos National Laboratory, Los Alamos, NM

ABSTRACT

Features for debugging a multitasking program have been added to the Instant Graphics Package (IGP) and to the dynamic debugging tool (DDT). These features include asynchronous breakpoints and multiple trace-back capabilities.

I. INTRODUCTION

This paper presents a brief overview of the debugging effort for multitasking at Los Alamos. Thus far, efforts have concentrated on providing fundamental software to link the debugging tool to the multitasking environment and avail the power of display and manipulation that a dynamic debugger normally allows in a uniprocess environment.

Basically, what we have done is to develop software links for multitasking, first for the Instant Graphics Package (IGP), which is a run-time graphics package that allows graphics and I/O to be decoupled from a code and processed either separately or concurrently. This software was then bridged to Livermore National Laboratory's dynamic debugging tool (DDT), giving us two forms of multitasking debuggers.

Dynamic debugging in a multitasking environment will require far more diligence and systems sophistication because of multiple, non-deterministic, concurrent executing tasks and their associated CPU connectors/exchange packages.

II. MULTITASKING COMMANDS

The following commands were added, modified or expanded for IGP and DDT to meet our fundamental multitasking debugging requirements.

CONNECT
TASK
BKP
LIST
DA

1. CONNECT - switches CPU connectors and reads in the executing exchange package. Debugging can then be performed in a normal manner.

2. TASK - switches to specified task and finds associated CPU connector. Debugging can then be performed in a normal manner.

3. BKP - this command has been modified to permit task specific breakpoints.

4. LIST - this command has been expanded to include the following:

   a) LIST ERROR - scans flag field of each executing exchange package for errors.

   b) LIST CONNECT - displays CPU connectors and status.

   c) LIST TRACE - performs trace backs for base level routines.

   d) LIST TASK - displays all tasks created and their CPU connector names and status.

   e) LIST TIB - displays the entire task information block. (not fully implemented)

   f) LIST SEMAPHOR - displays semaphore bits.

5. DA - Displays symbol table attributes for a variable such as class, equivalence, task common, storage length, and number of dimensions.

III. OTHER SOFTWARE

The multitasking debugger has been modified to recognize the UNICOS symbol tables.

IV. FUTURE MULTITASKING DEBUGGING PLANS

1. Merge certain IGP graphics capabilities, such as, one and two dimensional slice plots, two dimensional contour plots, and three dimensional perspective plots, with DDT.

2. Research a watch feature that can detect memory reads/writes for a specific variable.

3. Develop as needed features.

4. Develop a static controllee analyzer based on information contained in the symbol table.

REFERENCES

1. Reference for the Instant Graphics Package (IGP), Los Alamos National Laboratory, December 1985.

2. Dynamic Debugging Tool (DDT), Livermore National Laboratory, July, 1981.

Operating System Support for Parallel Processing

Steve Reinhardt
Cray Research, Inc.

The operating system support necessary for parallel processing is largely the ability to support multiple tasks/processes/logical CPUs (choose one) in one user memory area. Most of this work involves swapping, aborting, and scheduling processes which happen to share one memory area. The actual calls to work with the multiple processes are reasonably straightforward once the groundwork is laid. The calls necessary for macrotasking are the ability to create an extra process (tfork in UNICOS, TASK$CRE in COS), delete a process (exit, TASK$DEL), deactivate a process (pause, TASK$DEA), and activate a process (signal, TASK$ACT).

Once the support for macrotasking is in place, the extensions to support microtasking are minor. A quick review of the goals of microtasking may make the reasons for the OS support clearer.

User Goals

(a) Microtasked code should be nearly as efficient on one processor as the original single-processor code. Minimal code should be added to cater to additional processors.

(b) Microtasked programs should run as well in a mix as in a dedicated system. User code should dynamically use any available processors. It should not depend on a specific number of processors to complete work or to synchronize. Therefore, the leading processor should never wait for trailing processors.

(c) The implementation should generate the minimum number of additional system calls. Specifically, the user should not call the system to resolve inter-CPU data dependencies because data dependencies induced by an OS disconnect are infrequent enough to be resolved by the OS itself.

System Goals

(a) The most efficient way to use N CPUs is with N separate programs.

(b) If extra processors exist, they should go to efficient programs (those with highest percentage of parallelism).

(c) The system should detect when any program has a data dependency on a processor which has been removed and resolve the dependency.

(d) System scheduling even for microtasked programs should be simple. Connecting all processes from a program at the same time does not help system performance. The above user goals achieve high efficiency without simultaneous connection.

Meeting the Goals

The lead processor should never slow down for a chance of getting multiple processors. This implies several things.

(a) The lead processor cannot notify the system when it enters a parallel section of code. The overhead to get the extra processors is too big for the size of the average loop.

(b) Similarly, the lead processor cannnot notify the system when it leaves a parallel section of code. Gauging how long to hold processors before releasing them to the system is impossible in practice. The system should determine when a processor is no longer being used.

The XMP implementation works by coordinating the shared register/semaphores, the deadlock interrupt, and the wait-semaphore counter of the performance monitor. The lead processor puts loop indices in shared B/T registers instead of local B/T registers so other processors have equal access to remaining work. The lead processor uses several semaphores to signal any other processors when parallel sections of code are entered. Any waiting processors (which were waiting on a semaphore) would then work on the loop. At the end of the parallel section, the extra processors go back to a "park" point and wait on a semaphore. The lead processor continues through a non-parallel section.

OS Support for Microtasking

The two main pieces of support for the operating system are deadlock resolution and efficiency monitoring.

Two kinds of deadlocks exist. The classic deadlock is a programmer error in which all processes are unable to proceed. We call this a "hard" or "true" deadlock. A "soft" or "transient" deadlock occurs when a processor is removed from a program when it is holding a semaphore.

The deadlock interrupt notifies the system that all connected CPUs for a program cannot proceed and that some other process should be given the CPU. In a mix, a processor is commonly preempted by the OS. If the lead processor is removed in a non-parallel section, all the waiting processors immediately deadlock. If any processor is removed in the middle of a parallel section, the other processors will finish the remaining work of the parallel section. If the removed process returns before the others finish, all proceed normally, but if not the others will eventually wait for it. The deadlock signals that they have a data dependency on a process removed by the OS. The deadlock cases all result from OS intervention, which the OS resolves by rescheduling the deadlocking process after the disconnected process. We call these deadlocks transient deadlocks.

Efficiency monitoring is necessary because a running program never signals for more or fewer processors, so the system doesn't know which processors are really needed and which ones are expendable. With many CPUs and many programs, the system should pull processors away from inefficient programs and add them to efficient programs. On the XMP the wait-semaphore counter of the performance monitor provides a mechanism by giving an efficiency index. A process' priority is penalized for time spent waiting on a semaphore. The wait-semaphore time also allows consistent billing; thus, whether a microtasked program runs in a mix or by itself, it will always be charged the same amount.

PERFORMANCE I, I/O

Mostyn Lewis
Chevron Oil Field Research Company
La Habra, California

Three people spoke during this session:

1. Dan Cummings of Cray talked on I/O Performance Improvements. He covered enhancements made in the COS 1.15 release. These spanned changes to IOS, CIO/TIO in COS, and support for the Model C IOP.

2. Tony Shober of ATT analyzed a "schizophrenic" X-MP. A write-up of his talk follows after this item.

3. Jim Harrell of Cray went through changes to UNICOS made for performance purposes. A write-up of his talk follows after this item.

We extend thanks to the above for their work in preparing their talks and their well-received presentations.

Experiences with a Schizophrenic X-MP: A Look at X-MP Performance and Comments on the Coming of UNICOS

R. A. Shober

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

The first part of this paper reviews the performance impact of new X-MP hardware and FORTRAN compiler features; such as Gather/Scatter and the vectorization of IF statement constructs. Programming examples are taken from recently published supercomputer newsletters, and the results obtained from those examples using "plain" FORTRAN using CFT 1.14 and CFT 1.15 are presented. The second part of the paper presents some personal observations regarding the state of development of UNICOS for the X-MP, with emphasis on the user perspective.

1. INTRODUCTION

This paper is divided into two sections. The first section considers some performance implications on the X-MP using the CFT 1.14 and CFT 1.15 compilers. The second section reviews experience to date with UNICOS on our X-MP/24, with an emphasis on the user's perspective of UNICOS.

2. PERFORMANCE IMPACT OF NEW COMPILER FEATURES

This section reviews four examples of FORTRAN operations whose performance has been improved through recent CFT compiler updates. These examples are:

- Scatter/Gather
- Computed GO TO
- IF-THEN-ELSE IF
- Sparse Matrix Code

These examples were inspired by References 1-3. References 1-3 deal with the use of library routines to utilize the special features of the X-MP hardware. Recent compiler releases (CFT 1.14 and CFT 1.15) have made significant improvements in vectorizing FORTRAN code, and allow the FORTRAN user to access many of the same features from FORTRAN directly.

2.1 Scatter/ Gather

The CRAY-1 and early Cray X-MP supercomputers did not have a "hardware scatter/gather" feature; that is, the ability to load or store vector registers from random locations within central memory. The X-MP series of supercomputers now supports hardware scatter/gather, and the software in CFT 1.14 or 1.15 allows the user to take advantage of this hardware. Consider four different ways to program a "SPAXPY"* operation:

- Case A - Gather-Saxpy-Scatter (SCILIB).

  Perform discrete Gather, Saxpy, and Scatter operations using three distinct calls to the SCILIB routines GATHER, SAXPY, and SCATTER.

- Case B - Gather-Saxpy-Scatter (FORTRAN).

* A SPAXPY operation has the following form:

```
       do 10 i= 1,n
    10 a(j(i))= a(j(i))+ b*c(i)
```

Perform discrete Gather, Saxpy, and Scatter operations using three distinct FORTRAN DO loops.

- Case C - Spaxpy (SCILIB).

  Perform one call to the SCILIB routine SPAXPY.

- Case D - Spaxpy (FORTRAN)

  Perform one FORTRAN DO loop to accomplish the Spaxpy operation directly in FORTRAN.

For this example, consider the performance on the X-MP of these four cases. The results (performance measured in megaflops) are given in Table 1. Cases A and B (the GSS columns) are not optimal for the X-MP. Before the hardware scatter/gather feature was available on the X-MP, Case A was the only way to get any amount of vectorization for a SPAXPY operation; only the middle SAXPY operation was vectorized, while the SCATTER and GATHER executed assembler language code in scalar mode. On the X-MP with hardware scatter/gather, Cases 3 and 4 exhibit the best performance, and show equivalent asymptotic performance for long vector lengths, however for the other vector lengths shown the FORTRAN results have better performance than the (presumably assembler language) SCILIB SPAXPY routine. To understand this, it must be remembered that the comparison is between *in-line* FORTRAN code and *out-of-line* (i.e., using a subroutine call) assembler language. The overhead of the subroutine call and arguments adds between 1.5 and 2.5 microseconds to the execution time of the operation; these operations are sufficiently fast such that this overhead time is significant in reducing the effective performance of the SCILIB SPAXPY routine. (These results are essentially identical when using CFT 1.14 or CFT 1.15.)

TABLE 1 - SCATTER/GATHER EXAMPLE - RESULTS (MEGAFLOPS)

n	CASE			
	GSS (SCI)	GSS (FORT)	SPAXPY (SCI)	SPAXPY(FORT)
10	4	7	11	15
25	10	14	24	31
64	21	26	41	54
100	25	30	45	58
500	45	43	68	74
10000	54	47	78	79

From these results, the Gather-Saxpy-Scatter cases should not be used on the X-MP. The Saxpy operation written in FORTRAN yields the best performance as it does not suffer from the penalty of subroutine linkage overhead. The asymptotic performance in megaflops of a CFT FORTRAN Spaxpy operation is 79 megaflops (rather than the 55 megaflops quoted in Ref. 2).

2.2 Computed GO TO

The Computed GO TO example is shown in Figure 1. This Figure shows the original implementation of a loop using a computed GO TO statement, as well as a revised implementation using IF statements. The results for the original and revised implementations are shown in Table 2, with the "CFT" results using CFT 1.14 or CFT 1.15 (results identical for each). The results in Ref. 1 were obtained using assembler language library routines.

FIGURE 1 - COMPUTED GO TO EXAMPLE

Original Implementation	Revised Implementation
do 10 i= 1,n	do 20 i= 1,n
go to (1,2,3),itype(i)	if(itype(i).eq.1) then
1 z(i)= a*x(i)+ y(i)	z(i)= a*x(i)+ y(i)
go to 10	end if
2 z(i)= b*y(i)+ x(i)	if(itype(i).eq.2) then
go to 10	z(i)= b*y(i)+ x(i)
3 z(i)= x(i)+ y(i)	end if
w(i)= x(i)-y(i)+ c*z(i)	if(itype(i).eq.3) then
10 continue	z(i)= x(i)+ y(i)
	w(i)= x(i)-y(i)+ c*z(i)
	end if
	20 continue

TABLE 2 - COMPUTED GO TO - RESULTS

	Microseconds Per Iteration (n= 1000)	
	Original Implementation	Revised Implementation
Reference 1	1.5	0.15
CFT	1.3	0.12

Results in Table 2 show the CFT results (using either CFT 1.14 or 1.15) to be better than the Reference 1 results (which used CFT 1.13). In addition, special routines were not required for CFT to vectorize this loop.

2.3 IF-THEN-ELSE IF

The IF-THEN-ELSE IF example is shown in Figure 2. This Figure shows the original implementation of a loop using an ELSE IF statement, as well as a revised loop using IF statements. The results for the original and revised implementations are shown in Table 3.

FIGURE 2 - IF-THEN-ELSE IF EXAMPLE

Original Implementation	Revised Implementation
do 10 i= 1,n	do 20 i= 1,n
if(x(i).gt.y(i)) then	if(x(i).gt.y(i)) then
z(i)= sin(x(i))	z(i)= sin(x(i))
else if(x(i).lt.y(i)) then	end if
w(i)= cos(z(i))	if(x(i).lt.y(i)) then
end if	w(i)= cos(z(i))
10 continue	end if
	20 continue

TABLE 3 - IF-THEN-ELSE IF - RESULTS

	Microseconds Per Iteration (n= 1000)	
	Original Implementation	Revised Implementation
Reference 1	2.7	0.35
CFT	2.3	0.33

Results in Table 3 show the CFT results (using either CFT 1.14 or 1.15) to be better than the Reference 1 results (which used CFT 1.13). As mentioned above, special routines were not required for CFT to vectorize this loop.

It should be pointed out that the above technique of dividing a complex IF test into several individual IF tests works as long as the various tests are orthogonal. If the tests were not orthogonal, the results using the different implementations may not be identical.

2.4 Sparse Matrix Code Example

The Sparse Matrix Code Example is shown in Figure 3. This Figure shows the original implementation of two loops - an inner (dot product) loop and an outer loop - and a revised implementation patterned after the implementation in Ref. 1. The results for the original and the revised implementations are shown in Table 4.

FIGURE 3 - SPARSE MATRIX CODE EXAMPLE

Original Implementation

```
   do 9 i= 1,n
     ni= num(i)
     iss= istart(i)
     iee= iss+ ni-1
     smm= 0.0
     do 5 k= iss,iee
       smm= smm+ x(k)
 5   continue
     z(i)= smm
 9 continue
```

Revised Implementation

```
    do 100 ni= 1,9
    nni= 0
    do 10 i= 1,n
    if(num(i).eq.ni) then
      nni= nni+ 1
      list(nni)= i
    end if
10  continue
    do 11 i= 1,nni
      sum(i)= 0.0
      is(i)= istart(list(i))-1
11  continue
    do 20 k= 1,ni
    do 22 i= 1,nni
      sum(i)= sum(i)+ x(is(i)+ k)
22  continue
20  continue
    do 23 i= 1,nni
23    z(list(i))= sum(i)
100 continue
```

TABLE 4 - SPARSE MATRIX CODE EXAMPLE - RESULTS

	Microseconds Per Iteration (n= 1000)	
	Original Implementation	Revised Implementation
Reference 1	3.8	0.6
CFT 1.14	2.6	3.9
CFT 1.15	2.5	0.6

Results in Table 4 show that the CFT 1.14 result for the Revised Implementation is very poor. This is due to the "DO 10" loop in the Revised Implementation of Figure 3 that does not vectorize under CFT 1.14. However, when the CFT 1.15 compiler is used, this loop does vectorize and the results are comparable to the Reference 1 results.

2.5 Summary of Above Examples

When Supercomputers first appeared on the scene, compilers were limited in their ability to take full advantage of the vector hardware available. Special purpose libraries, such as Cray's SCILIB and Boeing's VectorPak served valuable roles to help the user take as much advantage as possible of the system. Recent improvements in the CFT compiler make the programmer's job much easier, as more standard FORTRAN constructs will vectorize. The examples above are illustrative of this point.

An even more interesting question is the role of such special purpose libraries in application program development today. Unquestionably, examples exist today where special purpose assembler language routines can substantially out-perform standard FORTRAN. For example, operations such as reduction operations (dot products) and linear recurrences either do not vectorize well or do not vectorize at all, thus the assembler language versions are clearly superior. The real question here is the issue of transportability of software when special purpose libraries are used. It is common for programs to be executed on a Cray computer, and as well on other computers such as mainframes, minisupercomputers, or superminicomputers. In such cases, it would be necessary to install the special purpose libraries on every system the program would be executed. Not only is this costly (if the library had to be purchased externally), but requires staff time to maintain the libraries, install bug fixes and new releases, and keep the versions synchronized. It is therefore unclear if the performance improvement gained from the special purpose libraries is worth the additional effort and expense if the programs must be transportable across different computing environments. As each organization performs computing in a somewhat different way, these issues must be considered with the local situation in mind.

3. UNIX ON THE X-MP - EXPERIENCE FROM THE USER PERSPECTIVE

The UNICOS system at AT&T Bell Laboratories is running under the Guest Operating System (GOS). Therefore, one processor is running COS in a batch environment with the other processor running UNICOS interactively. Some user level software is still not available - such as the "vi" editor - but with the standard UNICOS 1.0 recently installed such tools could be brought up. As a "C" machine, the system seems to be 25 times faster than a VAX 11/750 - but more than 25 times more expensive than a 750. Thus it appears that the real strength of the system will remain scientific number crunching, as it always has been.

Specific problem areas are:

- Scheduler - the system seems to run slow with several large interactive jobs in the system at once without an SSD for swapping.
- SEGLDR:
  - — Long execution (CPU) times.
  - — SEGLDR constructs an image of the a.out file in memory and then writes that image to disk. Blocks of storage (DIMENSIONed arrays) have explicit storage reserved in the a.out file. This causes lots of disk space used up to store blank words, and lots of time to roll in such a large image when execution begins.
- Debugger - poor information on where programs fail - sometimes the line number where the failure was supposed to be is wrong.

As an example of the above concerns, timing comparisons for several systems were done for a small problem. The problem is to solve the matrix system Ax= b where A is a dense 500x500 matrix. Standard LU factorization and solution methods were used by the routine "GELE" from the PORT3 library (4). The times for different systems are given in Table 5.

TABLE 5 - TIMING COMPARISON FOR FORTRAN PROGRAM

	SYSTEM			
	X-MP COS	X-MP UNIX	3081K UNIX	VAX 11/785 UNIX
Compiler Used	CFT	CFT	F77	F77
Execution Time (CPU sec.)				
Compiler	.02	.04	1.5	2.3
Load	.4	4.0	1.1	7.0
Execute	1.09	1.1	160	624

The data in Table 5 show that the compilation and execution times under COS and UNIX for the X-MP are essentially identical; the differences are due to the timing routines used rather than real differences in performance. The SEGLDR under UNIX, however, has poor performance in that it requires 10 times more CPU time to execute than the loader under COS. In addition to this, the SEGLDR required at least 26 seconds of wall clock time regardless if there were only one user on the system or more than one user. When the program was executed, it required 3 seconds to "roll in" the program from the time you type "a.out" until the time the program is entered and begins to run. These long times seem to be due to the length of the "a.out" image on disk resulting from the problems noted above. This test problem required about 288K words of memory. If the test problem were increased such that A was 1000x1000, the a.out file would be over one million words long and would require at least 20 seconds to roll the file into memory (regardless of the number of users on the system).

In discussing these problems with Cray staff, there is reason to hope that Cray is aware of the problems and is in the process of addressing them. There are also reasons to believe that performance on native UNICOS will be better than on UNICOS under GOS.

REFERENCES

[1] "Supercomputer Forum", Boeing Computer Services, November-December, 1985, pp 3-4.

[2] "Supercomputer Forum", Boeing Computer Services, January-March, 1986, pp 9-11.

[3] "Gather/Scatter", San Diego Supercomputer Center, March, 1986, pp 7-8.

[4] P. A. Fox, Ed., "The PORT III Mathematical Subroutine Library", AT&T Bell Laboratories, May 8, 1984.

Unicos Performance Enhancements

Jim Harrell

Unicos Development
Cray Research, Inc.
Mendota Heights, Minnesota 55120

Introduction

Performance is a very broad topic. This is especially true since we are talking about a new operating system, Unicos, that has run on other kinds of hardware. Unicos is based on System V Unix.[1] There are a number of design issues and architectural issues that are important and new. The range of issues includes: comparisons of the other machines that run the Unix operating system, the architectural issues of multi-CPU and multitasking implementations, and very simple things like disk block size changes.

Today I want to cover parts of one area of interest, I/O enhancements. We will look at some of the enhancements we've made to the Unicos system, some of the issues involved, and current performance. The information represents work done on the XMP series of machines but should not imply differences from Cray 2 Unicos. Differences occur for architectural reasons in the kernel, but the user levels are the same. Through all of this we will be looking at performance numbers that I refer to as "achievable" rates. That is these numbers were taken from a current unmodified system on a quiet Cray XMP.

1.0 I/O Performance

The original port of Unix to a Cray machine did not modify the basic structure of Unix I/O. Changes were generally made for architectural reasons. The disk block size is a good example; it was increased from 512 bytes to 4096 bytes. This was possible because (like the rest of Unix) the structure of the I/O handling is basically sound. It is small, and therefore fast. However, under this early system there were a number of problems from a Cray performance standpoint. Disk files could not span devices, and no "striping" was possible. Remember that performance is not just how fast can data be moved but also how much data can be read or written. The disk allocation scheme for files did not attempt to keep blocks within a file together so that files could potentially be scattered across a filesystem. This is discontiguous allocation. Contiguous allocation is important in achieving high disk transfer rates. There was no asynchronous I/O support in System V which is important to allow computation to continue during I/O transfers. The Unix disk buffer cache mechanism was extremely suspect, because for Cray usage the extra memory copy seemed not in the best interests of performance.

1.1 Spanning and Striping Disk Devices

Allowing files to span disk devices was solved by changing the minor device number to point to a new structure that contains a number of partitions instead of just one. This was a simple change without repercussions through the system. Disk striping works because of the I/O Subsystem (IOS). By carefully allocating partitions at the Unix configuration level, and notifying the IOS, striping worked. As an aside, the IOS has proved to be very useful during the move from COS to Unix. The low level work that it does simplifies many tasks. The change was simple, and it solved the problem. Remember it is important that we haven't modified the user interface in an incompatible manner, just added capabilities.

1.2 Contiguous Disk File Allocation

Disk allocation changes can be broken into two distinct phases. The first was done to achieve disk streaming rates. In part this was experimentation with Unix to see if high disk throughput was possible

[1] Unix is a trademark of AT&T.

simply. The second phase changed the allocation method for the disk files. While the second phase took longer neither were significant in terms of time to implement. The phases also neatly fit the releases. Phase 1 is release 1.0 of Unicos. Phase 2 is release 2.0.

1.2.1 Initial enhancements to achieve disk streaming rates

A set of experiments were done changing the "block" size from one sector to one entire cylinder. This sets the default size of any disk file to at least the "block" size. The cylinder transfers are exciting. The testing was very simple to do. The filesystem with the target block size was created, and a simple user program reads and writes the disk. Unix is a very good for measuring performance. User programs can be used in most circumstances, and timings are available from a program (time) called to monitor the user program. The results showed that streaming rates were possible for single devices even at four sector allocation levels. The performance at the four sector level was surprising, because the size was so small. The four sector size works because fewer requests to the IOS were necessary than at the 1 sector allocation, and the allocation size was adequate to keep up with the disks. However, track and cylinder allocation did give better overall performance, with more than one device. The downside is that performance is paid for with more limited disk utilization. Slide 8 shows the transfer rates for large block filesystems.

1.2.2 Contiguous Disk File Allocation Enhancements

File allocation under standard Unix does not attempt to allocate space on disk contiguously. Initial allocation might be contiguous but after many files have been deleted and created files tend to be extremely fragmented. This is a problem because in order to achieve maximum throughput blocks within files must be allocated contiguously. This must be for every file every day. This is a simple method to achieve disk throughput. The original enhancements to allow multiple disk filesystems forced files to be allocated in the first partition until that was full, and then into the second, and so on. This worked best with small partitions but did not use the multiple disks wisely. The new scheme uses a bit map to force contiguous allocation, and to put files on different devices if at all possible. Moreover because the contiguous allocation speeds transfers the large block file systems are no longer necessary. The speedup was visible. The transfer rates were up and the system overhead was down. Slides 10 through 13 show the transfer rates and system overhead for contiguous allocation enhancements.

1.3 Asynchronous I/O

One of the familiar features needed in System V was asynchronous I/O. This feature was added in two ways. The first was to port the reliable COS style "listio". This provided a well known interface for COS programs that used asynchronous I/O. The second method added is the Cray2 "reada/writea" mechanism. The third method of handling asynchronous I/O is with the BSD "select" system call. This is the only method indigenous to Unix of handling asynchronous I/O. Currently the select system call is available through the Wollengong TCP/IP package. These three mechanisms each provide the user with the ability issue I/O requests and continue computing.

1.4 Unix Disk Cache

Standard Unix provides a set memory buffers for use as an I/O cache. The cache holds much valuable information in terms of directories and inode blocks which point to files on disk. In addition, if at all possible, the data from pipe transfers are also held in the cache. This is significant for interprocess communication. That is, the way that two programs communicate to each other. This is a wonderful little feature that if you don't already know about is well worth investigation. The cache also allows a process to read and write, reread and rewrite disk data before it ever leaves memory. The performance advantages are obvious. For short transfers, those that can remain in cache, I/O bandwidth is not tied to disk speeds. The size of cache is of course one limiting factor. Cache size must be large enought to provide room for necessary data but not so big as to restrict the size of programs unnecessarily. The wildcard in cache performance is the amount of simultaneous usage by other processes. Slides 16 and 17 show cache transfer rates for a simple test that overflows the disk cache using various increments in request size. What this shows is that the write transfers are faster. The program writes, and then reads. The writes can complete independently of the actual I/O. The reads must wait for some of the data to be fetched from disk, although the cache does hold most of the file. Thus the reads in this test are slightly slower. The write rates are actually faster than the theoretical maximum for DD49's. But they are still relatively slow because the actual requests must be satisfied one 4k block at a time. This accounts for the flat rates. The second set of tests, shown in slide 18, are for a short file of 200 sectors. The rates are substantially higher because the cache is not over filled. The extra figure between the write and read rates is the second run effect. Usually repeating a test ensures that the rates are correct. Since the file is in cache much of the early set up processing is bypassed. The rate shows how much the set up time effects the transfer rates. One other note about the numbers is that the drop in system overhead from release 1.0 to release 2.0 is attributable to the allocation enhancement changes discussed earlier.

1.5 Other "Disk Devices"

Some other interesting test results come from using various devices as "disk" in Unix. This is again very simple to do, and very easy to test. The slides 21, 22, and 23 show rates for BMR, SSD, and pipes.

slide #8

Disk Streaming

- "Large Block" filesystems

release 1.0, 1 stream, raw I/O, 1000 sectors
transfer rates in megabytes

sectors/req	1	4	track	cylinder
write	5.84	9.1	9.4	9.7
read	5.35	8.9	8.7	8.5

No measurements for release 2.0

slide #10

Contiguous File Allocation

release 1.0, 1 stream, raw I/O, 2000 sectors

sectors/req	1	4	8	32
write	5.2	5.3	5.5	5.9
read	5.1	5.8	5.8	5.5

release 2.0, 1 stream, raw I/O, 2000 sectors

sectors/req	1	4	8	32
write	5.6	9.5	9.7	9.6
read	5.2	9.2	9.2	9.2

slide #11

Contiguous File Allocation

release 1.0, 4 stream, raw I/O, 2000 sectors
transfer rates in megabytes

sectors/req	1	4	8	32
write	2.8	3.3	3.4	3.6
read	2.7	3.2	3.4	3.4

release 2.0, 4 stream, raw I/O, 2000 sectors
transfer rates in megabytes

sectors/req	1	4	8	32
write	3.5	8.1	9.5	9.5
read	3.4	7.6	9.4	9.3

slide #12

Contiguous File Allocation

release 1.0, 1 stream, raw I/O, 2000 sectors
times in seconds

sectors/req	1	4	8	32
elapsed	3.15	2.98	2.96	2.93
user	0.13	0.03	0.02	0.005
sys	0.86	0.72	0.70	0.67

release 2.0, 1 stream, raw I/O, 2000 sectors
times in seconds

sectors/req	1	4	8	32
elapsed	3.29	1.75	1.75	1.75
user	0.05	0.02	0.008	0.003
sys	0.76	0.17	0.09	0.02

slide #13

Contiguous File Allocation

release 1.0, 4 streams, raw I/O, 2000 sectors
times in seconds

sectors/req	1	4	8	32
elapsed	6.31	5.15	4.88	4.77
user	0.50	0.18	0.14	0.049
sys	1.21	0.94	0.87	0.77

release 2.0, 4 streams, raw I/O, 2000 sectors
times in seconds

sectors/req	1	4	8	32
elapsed	4.86	2.38	1.97	1.98
user	0.16	0.03	0.014	0.003
sys	1.19	0.26	0.099	0.026

slide #16

Unix Disk Cache

release 1.0, 1 stream, 2000 sectors
transfer rates in megabytes

sectors/req	1	4	8	32
write	4.4	4.1	4.2	3.9
read	3.2	3.4	3.4	3.2

release 2.0, 1 stream, 2000 sectors
transfer rates in megabytes

sectors/req	1	4	8	32
write	11.7	11.7	11.7	11.4
read	5.9	7.6	7.8	7.9

slide #17

Unix Disk Cache

release 1.0, 1 stream, 2000 sectors
times in seconds

sectors/req	1	4	8	32
elapsed	4.60	4.60	4.53	4.77
user	0.14	0.04	0.02	0.006
sys	0.80	0.63	0.61	0.59

release 2.0, 1 stream, 2000 sectors
times in seconds

sectors/req	1	4	8	32
elapsed	2.29	2.00	1.97	1.95
user	0.06	0.02	0.01	0.003
sys	0.66	0.54	0.52	0.50

slide #18

Unix Disk Cache

release 1.0, 1 stream, 200 sectors
transfer rates in megabytes

sectors/req	1	4	8	32
write	15.8	59.6	64.7	69.1
	29.3			
read	34.8	48.8	51.9	54.3

release 2.0, 1 stream, 200 sectors
transfer rates in megabytes

sectors/req	1	4	8	32
write	39.9	91.2	115.9	121.4
	49.3			
read	42.8	70.1	81.2	83.9

slide #21

Other "Disk" Devices

• BMR

No measurements available for release 1.0

release 2.0, 1 stream, raw I/O, 400 sectors
transfer rates in megabytes

sectors/req	1	2	4	8	32
write	7.0	12.3	17.8	22.9	30.2
read	7.5	12.7	19.2	25.8	34.8

• SSD

release 1.0, 1 stream, async I/O, 1.7MW, 128MW SSD
transfer rates in megabytes

sectors/req	4	18	36	1864
rate	64.78	260.98	468.57	1974.32

No measurements available for release 2.0

slide #22

Other "Disk" Devices

- Pipes

rel 2, 1 connection, 1000 transfers
transfer rates in megabytes

sectors/req	1	2	4	8
write	18.7	26.5	31.5	34.4
read	18.6	26.4	31.4	34.3

rel 2, 1 connection, 1000 transfers
transfer rates in megabytes

sectors/req	1	2	4	8
write	24.3	35.8	44.5	46.9
read	24.1	35.6	44.3	46.9

slide #23

Other "Disk" Devices

- Pipes

rel 2, 1 connection, 1000 transfers
times in seconds

sectors/req	1	4	8	32
elapsed	0.30	0.41	0.59	1.12
user	0.11	0.17	0.15	0.10
sys	0.08	0.12	0.22	0.44

rel 2, 1 connection, 1000 transfers
times in seconds

sectors/req	1	4	8	32
elapsed	0.22	0.34	0.46	0.87
user	0.01	0.01	0.01	0.01
sys	0.07	0.10	0.17	0.33

NETWORKING SESSION I

Dean W. Smith

ARCO Oil and Gas Company
Plano, Texas

This session consisted entirely of a number of talks by Cray Research personnel responsible for the development and support of Cray's station products. Presentations were made on Superlink and a Multiple Frontends Support Facility. This report is based on my notes and overheads provided.

SUPERLINK 2.0

The Superlink 2.0 status presentations were made by Stewart Ross (Cray Research U.K.), and Brian Gaffey and Nic Catrambone of Cray Research Inc., Mendota).

The Superlink product family is designed to meet the expanding application and data accessibility needs of the super-computing community. Providing a high performance data pipe, new external interface for applications, standard data access, and access of all peripherals of the host system.

The stated direction of Superlink is 4 fold: 1) to provide an OPEN systems approach based on ISO OSI, 2) common interface for all applications, 3) alignment with industry standards, 4) commitment to both performance and functionality.

The current Superlink product is Superlink/ISP release 1.0 and is available for the COS/MVS environments. Superlink 2.0 is scheduled for availability in the COS/MVS environments in late 1986, UNICOS in 1987, COS support in COS 1.16. Superlink 2.0 beta testing is scheduled for year end 1986.

Multiple Frontends Support

The problem addressed by Multiple Frontends Support (MFS) is that of identifying a user and their CRAY job's privileges across any number of frontends (i.e. a job submitted from a work station might require datasets from a VAX, tape support from an IBM, and have the printed output sent to another system). This problem is being seen increasingly as more and more sites try to provide all components of their system's networking resources to CRAY jobs. MFS is seen as a system which can address and resolve the conflicts inherent in networking jobs across multiple systems.

MFS, as it is being conceived, will provide user oriented information that will enable the frontend environments to resolve the conflicts that arise in a networking environment (i.e. what identifies a VAX user and their security privileges to an IBM database - and whose information can a frontend system trust). MFS will accomplish this by maintaining a database of user characteristics on the CRAY and provide those characteristics to frontends as services are required.

The development for MFS is by no means complete and the parties within Cray Research responsible for its development encourage any input on requirements.

NETWORKING SESSION II

Dean W. Smith

ARCO Oil and Gas Company
Plano, Texas

This session consisted entirely of a number of talks by Cray Research personnel. Presentations were made on TCP/IP support and the Multiple Protocol Support (USCP).

Bryan Krok's (Cray Research Inc., Mendota) report on his presentation of TCP/IP support follows this report. A report on Tricia Senn's presentation is based on my notes of her presentation and her overheads.

Multiple Protocol Support

Tricia Senn's report on Multiple Protocol Support detailed the Station Call Processor (SCP) of UNICOS. The UNICOS Station Call Processor (USCP) implementation will be critical providing a smooth migration path to UNICOS in existing systems.

The current features supported by USCP are dataset staging, interactive, job submission, and job status and control. Features of SCP not yet supported are system operator commands and tape message support.

An important feature of USCP support will be the degree of transparency that user's will experience using USCP and their experiences with SCP. In this regard, the JCL interfaces of USCP will be very familiar to the user of SCP. USCP's provided FETCH, ACQUIRE, and DISPOSE statements are all much the same as SCP's statements. However, program call interfaces will not be supported.

Supported dataset formats include SCP's supported formats of CB, BB, TR, CD, and BD. Additional support for UD (UNICOS data) has been included.

Supported operator commands include: ENTER (modifies jobs attributes), DROP (cancels an active job), ROUTE (changes a job's routing), KILL (cancels a job or output), and STREAM (modifies a stations staging attributes).

Hardware support will be provided for NSC and FEI interfaces.

Communications III - User Presentations

Ronald Kerry

General Motors Research Labs

This session consisted of two user presentations followed by an open forum for discussion of communications issues. Brian Vohs of EXXON Corporation presented two topics: "Adding Commands to the MVS Station" and "JES3 Dependent Job Networking on a CRAY". Sandy Moy of the University of Illinois at Urbana-Champaign presented a talk on the "University of Illinois Network". The following sections are based on the notes and overheads from the individual presentations.

Adding Commands to the MVS Station

Brian Vohs
EXXON

EXXON processes many CRAY jobs through the MVS station. Often the job load is large enough to cause a backlog of jobs on the MVS front end, especially when the CRAY is not in service. The CRAY supplied MVS station provides no way to tell how many jobs are backlogged waiting for the station to reconnect to the CRAY. Therefore a new station command was written. Adding station commands to the MVS Station is a fairly simple process. It can be done using three modifications to the station (included at end of paper).

The first modification adds the command name to the CRAYCMD command processor used by TSO users.

The second modification adds the command name to the table of operator commands (CRCTABL). This table defines the name of the module which is used to process the command as well as whether or not the command can be accessed from TSO.

The final modification simply adds the command processor named above to the MVS Station. Before this can be done the module must be added as an entry point in MVS Station load module CSS001 using SMP UCLIN.

JES3 Dependent Job Networking on a CRAY

Brian Vohs
EXXON

EXXON uses the Dependent Job Control (DJC) facility of JES3 on their IBM front end machines extensively. This facility allows a user to specify the order of execution for multiple jobs. A set of such jobs is known as a DJC network. Each job in the network can have predecessor jobs and successor jobs. A job will not run until all of its predecessor jobs have completed successfully. When EXXON obtained the CRAY, this facility had to be extended so that CRAY jobs could be part of a DJC network.

The implementation of this function involves one modification to the EXP task of COS and two modifications to the MVS station support for JES3. The text for these modifications is included at the end of this paper.

In the future, EXXON plans to extend this concept so that a single job having multiple steps can have some steps run on the CRAY while other run on the front end. This facility is much more difficult to implement, but will give user even greater flexibility in the type of work that they can do.

University of Illinois Networks

Sandy Moy
University of Illinois at Urbana-Champaign

The University of Illinois at Urbana-Champaign is the home of the National Center for Supercomputing Applications (NCSA). NCSA is funded primarily by the National Science Foundation (NSF). The NCSA is a national center on which time is allocated through peer review by the NSF or NCSA Peer Review Board. It is a place where supercomputer users come together to learn, experiment, and exchange ideas about new research and new computational techniques. It is a training ground for graduate and post-doctoral students, faculty, and visiting researchers and serves as an interdisciplinary clearinghouse dedicated to creating new computational disciplines, eliminating common computational bottlenecks, and improving code efficiency. Supercomputer users from around the country are encouraged to spend some time at the center.

The center is supported by a comprehensive, networked computing environment including the latest in workstation capabilities, both hardware and software. Every office has one or more workstations for interactive work on the CRAY and for other research purposes such as editing, preparing proposals and papers, and graphic and image processing.

The NCSA CRAY has 2 CPUs, 4 million words of memory, a 32 million word solid state disk (SSD) and 7200 million bytes of online disk storage. This configuration will be upgraded during 1986 to a CRAY XMP/48 with a 128 Mword SSD and increased disk storage capacity. The operating system is CTSS, an interactive timesharing system developed to increase programming and scientific productivity in the supercomputing environment.

The front end machine is a VAX 11/785 which handles CRAY interactive traffic, provides a limited mail service for NCSA users, and acts as a temporary staging device for file exchange. The operating system is VMS. A second VAX 11/785 will be available in 1986 to form a VAX cluster to handle an increasing number of users.

The Common File System (CFS) is a hierarchical multimedia storage system used for file storage and archiving of data. It is supported by an IBM mainframe, a disk farm, a 55 billion byte mass storage system and 4 IBM cartridge tape drives.

The CRAY, the front end VAX, and CFS are connected via Network Systems Corporation's HYPERchannel, which has a bandwidth of 50 million bits per second. The VAX provides a gateway to the CRAY for the PROTEON campus fiber optic backbone network, a Vitalink satellite network and a series of 56 Kbps links to other national centers and campuses. Other remote network capabilities include INWATS, BITNET, Telenet and ARPAnet.

NCSA uses the TCP/IP protocol to interconnect workstations, the VAX and the CRAY. The CRAY Operating System (COS) had to be modified to support this interconnection. The reasons for this decision include:

* Existing networking protocols are chaotic
* Too much work required by front ends for protocol conversion
* TCP is a standard. NSF likes it. Furthermore, TCP provides a migration path to the future ISO protocols.
* TCP is widespread; virtually every UNIX machine in the world, including most workstations, understands it.
* TCP is more general purpose than NSP or SIMP.

Current CRAY network services include:

* Remote login to CTSS (local)
* File transport (local)
* File archival
* Standard DARPA services (local)

Services to be provided in the near future include:

* Remote login via TELNET
* File transport to anywhere via FTP
* VMS mail on CTSS via POP
* High speed file transport to the front ends and CFS via NETBLT/IP
* Ability to communicate arbitrarily and reliably to any machine on the network
* TCP/IP and friends on any machine, with optional transport level and presentation level gateways to other networks

NCSA networking goals include the integration and coordination of multiple networks, state of the art network services and flexibility for the future.

```
++USERMOD(MODID#1)          /*
                               ADD BLOG TO CRAYCMD COMMAND PROCESSOR IN
                               CRAY MVS WORKSTATION PACKAGE
                                                                  */ .
++VER (Z038) FMID(#CSS003) /* CRAY WORKSTATION PACKAGE V1.13 */ .
++SRCUPD (CRAYCMD) .
./ CHANGE NAME=CRAYCMD
         DC    CL11'BLOG',AL1(3),V(PCLNULL),A(PANULL)           MODID#1 11101000



++USERMOD(MODID#2)          /*
                               ADD BLOG TO TABLE OF COMMANDS IN
                               CRAY MVS WORKSTATION PACKAGE
                               COMMAND NAME = BLOG
                               MINIMUM LENGTH = 4
                               MODULE TO PROCESS COMMAND = CRXCRAY
                               FUNCTION CODE FOR MODULE = 0
                               COMMAND IS ACCESSIBLE TO TSO USERS
                                                                  */ .
++VER (Z038) FMID(#CSS003) /* CRAY WORKSTATION PACKAGE V1.13 */ .
++SRCUPD (CRCTABL) .
./ CHANGE NAME=CRCTABL
CCBLOG   CMD   BLOG,4,,MODULE=CRXBLOG,FC=0,TSO=YES              MODID#2 00311000




++USERMOD(MODID#3)          /*
                                COMMAND PROCESSOR TO OPERATE WITHIN THE
                                CRAY MVS WORKSTATION PACKAGE.
                                BEFORE THIS MODULE CAN BE ADDED TO THE
                                LOAD MODULE, THE FOLLOWING UCLIN MUST
                                BE RUN TO DEFINE THIS MODULE IN LOAD
                                MODULE CSS001:
  UCLIN CDS.
    ADD SRC(CRXBLOG) DISTLIB(CRAYSRC) FMID(#CSS003) .
    ADD MOD(CRXBLOG) DISTLIB(CRAYSRC) FMID(#CSS003) LMOD(CSS001) .
  ENDUCL .
  UCLIN ACDS.
    ADD SRC(CRXBLOG) DISTLIB(CRAYSRC) FMID(#CSS003) .
    ADD MOD(CRXBLOG) DISTLIB(CRAYSRC) FMID(#CSS003) LMOD(CSS001) .
  ENDUCL .

                               */ .
++VER (Z038) FMID(#CSS003) /* CRAY WORKSTATION PACKAGE V1.13 */ .
++SRC (CRXBLOG) .
```

```
*ID EXXEXP,DC=EXP
*/
*/   MODS TO EXP
*/
*/
*/   MOD TO SUPPORT CRAY/IBM JOB NETWORKING
*/
*I M06586KA.5873 AFTER LABEL 'TRM'
         GETF,S0    S7,JTABTC,A2   GET ABORT CODE
         JSN        SKPRESET       ABORTING - SKIP THIS MESS
         GETF,A3    S7,JTJXT,A2    GET POINTER TO JXT
         GETF,A7    S7,JXSDT,A3    GET OFFSET WITHIN SDT
         A3         B@SDT          GET POINTER TO SDT
         A7         A7+A3          POINT TO SDT ENTRY (OFFSET+BASE)
         GETF,A3    S7,SDTXT,A7    GET POINTER TO TEXT AREA
         GETF,A4    S7,SDTXC,A7    GET INCREMENT TO SLOT (TEXT LENGTH)
         A7         O'40           OFFSET TO 'NETJOB' IN SLOT
         A3         A3+A4          POINT A3 AT THE SLOT
         A3         A3+A7          POINT A3 AT 'NETJOB'
         S6         0,A3           PICK UP ' NETJOB '
         S7         1,A3           PICK UP NEXT WORD
         A7         D'8            AMOUNT FOR SHIFT
         S6         S6,S7<A7       SHIFT OFF 1 BYTE
         S7         NETJOBC,0      GET LITERAL 'NETJOB C'
         S0         S7-S6          GET DIFFERENCE
         JSN        SKPRESET       NO - SKIP RESET CODE
         S6         NETJOBD,0      PICK UP 'NETJOB D'
         S7         1,A3           RELOAD ORIGINAL WORD
         S7         S7<A7          SHIFT OFF THE 'C'
         S7         S6,S7>A7       OVERLAY THE 'C' WITH A 'D'
         1,A3       S7             SAVE IT BACK
SKPRESET =          *              COMMON EXIT POINT
*I M06586KA.6513
NETJOBC  CON        E'NETJOB C'                C'NETJOB C' IN EBCDIC
NETJOBD  CON        E'NETJOB D'                C'NETJOB D' IN EBCDIC
*/
*/       END OF MOD EXXEXP
*/
```

```
++ USERMOD (EXJ0105) /* THE FOLLOWING FUNCTIONS IS SUPPORTED:
                        CRAY WORKSTATION PACKAGE SUPPORT
                                                         */ .
++ VER (Z038) FMID(HJS2327)  /* JES3 SP1.3.1 $$$ XA $$$ */ .
++ SRCUPD (IATGRWQ) DISTLIB(AJES3SRC) .
./ CHANGE NAME=IATGRWQ
         TITLE 'MAP NCB - NETWORK CONTROL BLOCK'                  EXJ3S01 11372600
         IATYNET TYPE=DSECT,BLOCK=NCB                             EXJ3S01 11372700
         MVI   SJ3RLNET,C' '         SET IN CASE IT IS NOT A DJC JOB 3S01 14562410
         TM    JCTDJFL1,JCTDJJOB  IS IT A DJC JOB?                EXJ3S01 14562420
         BC    ALLOFF,JCTBLD2       NO - SKIP KEEPING THE NETID   EXJ3S01 14562430
         MVC   SJ3RLNET(L'JCTDJNET),JCTDJNET  KEEP NETID          EXJ3S01 14562440
JCTBLD2  DS    0H                                                 EXJ3S01 14562450
*-------------------------------------------------------------* EXJ3S01 14563805
*        EXTRACT FIELDS FROM THE JNCB/NCB AREA                  * EXJ3S01 14563810
*-------------------------------------------------------------* EXJ3S01 14563815
         CLI   SJ3RLNET,C' '         IS IT REALLY THERE?          EXJ3S01 14563817
         BC    EQ,JDABBLD            NO - GO ON AROUND THIS STUFF EXJ3S01 14563819
         JNCBHLD ID=SJ3RLNET,NORMAL=GOTJNCB GET/HOLD THE JNCB     EXJ3S01 14563820
         B     JDABBLD              GET OUT NOW - JNCB NOT FOUND  EXJ3S01 14563825
GOTJNCB  DS    0H                                                 EXJ3S01 14563830
         LTR   R3,R1                SAVE IT AGAIN (JUST IN CASE)  EXJ3S01 14563840
         BC    ZERO,JDABBLD         NOT A NET - GET OUT           EXJ3S01 14563842
         NCBTAFND ENTRY=(R3),NCB=SJ3JBNAM,NORMAL=FOUNDNCB         EXJ3S01 14563845
         B     RELJNCB              BAD - GO RELEASE THE JNCB     EXJ3S01 14563850
FOUNDNCB DS    0H                                                 EXJ3S01 14563855
         LTR   R1,R1                IS IT REALLY THERE?           EXJ3S01 14563857
         BC    ZERO,RELJNCB         NO - GO RELEASE THE HELD JNCB EXJ3S01 14563858
         USING NCBENTRY,R1          SET BASE FOR NCB              EXJ3S01 14563860
         MVC   SJ3RLJOB(L'NCBRLJOB),NCBVEND MOVE 1ST RELEASE JOBEXJ3S01 14563865
         MVC   SJ3ICTCH(8),=CL8'NETJOB C' MOVE IN ICATCHER/STAT EXJ3S01 14563867
         DROP  R1                   DROP BASE FOR NCB             EXJ3S01 14563870
         NCBTAREL ENTRY=(R3)        RELEASE THE NCB WE GO BEFORE  EXJ3S01 14563880
RELJNCB  DS    0H                                                 EXJ3S01 14563885
         JNCBREL ID=SJ3RLNET        RELEASE THE JNCB              EXJ3S01 14563895
*        THIS LINE DELETED BY EXJ3S01                             EXJ3S01 14568900
*        THIS LINE DELETED BY EXJ3S01                             EXJ3S01 14568950
*        THIS LINE DELETED BY EXJ3S01                             EXJ3S01 14569000
```

```
++USERMOD (EXJ0335)   /* CRAY STATION JES3 SUPPORT --
                         MODS TO IATCR29 TO SUPPORT NETWORKING
                         BETWEEN CRAY AND IBM
                                                              */ .
++VER (Z038) FMID(HJS2327) .
++SRCUPD (IATCR29) DISTLIB(AJES3SRC) .
         IATYNET TYPE=DSECT,BLOCK=NCB                          EXJ3S01 00016500
         SR    R4,R4               ... ZAP R4 FOR LATER        EXJ3S01 00026900
         CLC   SJ3ICTCH(L'NETJOB),NETJOB  IS IT NETJOB?        EXJ3S01 00027000
         BC    NE,NOTNETJB         NO - CONTINUE PROCESSING    EXJ3S01 00027100
         LA    R0,80               GET LENGTH OF GETMAIN       EXJ3S01 00027200
         AGETMAIN SIZE=(R0),BUSY=NOTNETJB IF BUSY - BYPASS ALL EXJ3S01 00027300
         LR    R4,R1               SAVE BUFFER ADDRESS         EXJ3S01 00027400
         XC    0(80,R4),0(R4)      CLEAR BUFFER OUT NOW        EXJ3S01 00027500
         MVC   1(L'MODNET,R4),MODNET BUILD '*F,N,ID=' PART     EXJ3S01 00027600
         MVC   L'MODNET+1(8,R4),SJ3RLNET MOVE IN NETID         EXJ3S01 00027700
         LA    R1,8                MAX LENGTH OF NETID         EXJ3S01 00027800
ENDJOBLP DS    0H                                              EXJ3S01 00027900
         LA    R9,L'MODNET(R1,R4) POINT TO END OF STRING       EXJ3S01 00028000
         CLI   0(R9),C' '          IS IT A BLANK?              EXJ3S01 00028100
         BNE   JOBLPOUT            NO - GET OUT                EXJ3S01 00028200
         BCT   R1,ENDJOBLP         LOOP FOR ALL 8 SPOTS        EXJ3S01 00028300
         B     NOTNETJB            BLANK NET ID - GET OUT      EXJ3S01 00028400
JOBLPOUT DS    0H                                              EXJ3S01 00028500
         MVI   1(R9),C','          MOVE IN A COMMA             EXJ3S01 00028600
         LA    R9,2(,R9)           POINT PAST THE COMMA        EXJ3S01 00028700
         CLI   SJ3STAT,C'D'        IS IT A DECREMENT?          EXJ3S01 00028800
         BNE   TRYNET@             NO - MUST FLUSH LATER       EXJ3S01 00028900
         MVC   0(L'ENDNET,R9),ENDNET BUILD 'J=XXXX,'           EXJ3S01 00029000
         LA    R9,2(,R9)           POINT TO THE XXXX           EXJ3S01 00029100
TRYNET@  JNCBHLD ID=SJ3RLNET,NORMAL=GOTJNCB  GET JNCB          EXJ3S01 00029200
         B     NOTNETJB            JNCB NT FND - NOT A NET     EXJ3S01 00029300
GOTJNCB  DS    0H                                              EXJ3S01 00029400
         LTR   R3,R1               SAVE IT AGAIN (JUST IN CASE) EXJ3S01 00029500
         BC    ZERO,NOTNETJB       NOT FOUND - GET OUT NOW     EXJ3S01 00029600
         CLI   SJ3STAT,C'D'        IS IT A DECREMENT?          EXJ3S01 00030000
         BNE   RELJNCB             NO - MUST RELEASE JNCB      EXJ3S01 00030100
         NCBTAFND ENTRY=(R3),NCB=SJ3RLJOB,NORMAL=FOUNDNCB      EXJ3S01 00030200
         B     RELJNCB             BAD - GO RELEASE THE JNCB   EXJ3S01 00030300
```

```
FOUNDNCB DS     OH                                             EXJ3S01 00030400
         LTR    R1,R1                 DID WE FIND ONE?          EXJ3S01 00030500
         BC     ZERO,RELJNCB          NO - GET OUT NOW          EXJ3S01 00030600
         USING  NCBENTRY,R1           SET BASE FOR NCB          EXJ3S01 00030700
         LH     R15,NCBJOBNO          GET JOB NUMBER TO BE RELEASED EXJ3S01 00030800
         DROP   R1                    DROP BASE FOR NCB         EXJ3S01 00030900
         CVD    R15,72(,R4)           CONVERT IT TO PACKED DECIMAL  EXJ3S01 00031000
         UNPK   48(15,R4),72(8,R4)    CONVERT IT TO ZONED DECIMAL   EXJ3S01 00031100
         OI     62(R4),C'0'           MAKE IT ALL PRINTABLE     EXJ3S01 00031200
         MVC    0(4,R9),59(R4)        MOVE RESULTS INTO MESSAGE TEXTEXJ3S01 00031300
         LA     R9,5(,R9)             POINT TO STATUS FIELD     EXJ3S01 00031400
         NCBTAREL ENTRY=(R3)          RELEASE THE NCB BEFORE WE GO  EXJ3S01 00031500
RELJNCB  DS     OH                                             EXJ3S01 00031600
         JNCBREL ID=SJ3RLNET          RELEASE THE JNCB          EXJ3S01 00031700
DOFLUSH  DS     OH                                             EXJ3S01 00031800
         MVC    0(1,R9),SJ3STAT       AND MOVE IN THE STATUS BYTE   EXJ3S01 00031900
         SR     R9,R4                 GET LENGTH OF DATA IN BUFFER  EXJ3S01 00032000
         STC    R9,0(R4)              SAVE LENGTH IN MESSAGE    EXJ3S01 00032100
NOTNETJB DS     OH                                             EXJ3S01 00032200
         LTR    R4,R4                  DID WE DO A GETMAIN      EXJ3S01 00041500
         BZ     UX29RET                IF NOT, GET OUT          EXJ3S01 00041600
         INTERCOM CONS=DUMMY,TEXT=(R4)                         EXJ3S01 00043100
         LA     R0,80                  GET LENGTH TO GIVE BACK  EXJ3S01 00043200
         APUTMAIN SIZE=(R0),AREA=(R4),SP=0                     EXJ3S01 00043300
         B      UX29RET                RETURN TO IATISEN        EXJ3S01 00043400
NETJOB   DC     C'NETJOB'             EYE CATCHER IN THE SLOT   EXJ3S01 00048000
MODNET   DC     C'*F,N,ID='           HEADER FOR MESSAGE        EXJ3S01 00048100
ENDNET   DC     C'J=XXXX,'            END OF MESSAGE            EXJ3S01 00048200
```

OPERATING SYSTEMS SESSION I

Raymond Benoît

Environnement Canada
Montréal, Canada

Three presentations were made in the operating systems session I which was devoted entirely to UNICOS. In the first presentation Denis Ritchie talked about Bell Labs experience in implementing and using UNICOS on their X-MP 24. Martin Fouts, from NASA Ames, then reported some preliminary findings and made some recommendations regarding UNICOS disk I/O on the Cray-2. In the last half hour of the session Matt Bishop, of NASA Ames, talked about Unix security issues and in particular those related to writing setuid programs.

I would to thank the speakers for participating in this OSC session and for the fine job they did in presenting their papers.

How To Write a Setuid Program

Matt Bishop

Research Institute for Advanced Computer Science
NASA Ames Research Center
Moffett Field, CA 94035

EXTENDED ABSTRACT

A typical problem in systems programming is often posed as a problem of keeping records [ALEP71]. Suppose someone has written a program and wishes to keep a record of its use. This file, which we shall call the *history file*, must be writable by the program (so it can be kept up to date), but not by anyone else (so that the entries in it are accurate.) UNIX† solves this problem by providing two sets of identifications for processes. The first set, called the *real* user identification and group identification (or UID and GID, respectively), indicate the real user of the process. The second set, called the *effective* UID and GID, indicate what rights the process has, which may be, and often are, different from the real UID and GID. The protection mask of the file which, when executed, produces the process contains a bit which is called the *setuid* bit. (There is another such bit for the effective GID.) If that bit is not set, the effective UID of the process will be that of the person executing the file; but if the setuid bit is set (so the program runs in *setuid mode*), the effective UID will be that of the owner of the file, not that of the person executing the file. In either case, the real UID and GID are those of the person executing the file. So if only the owner of the history file (who is the user with the same UID as the file) can write on it, the setuid bit of the file containing the program is turned on, and the UIDs of this file and the history file are the same, then when someone runs the program, that process can write into the history file.

These programs are called *setuid programs*, and exist to allow ordinary users to perform functions which they could not perform otherwise. Without them, many UNIX systems would be quite unusable. An example of a setuid program performing an essential function is a program which lists the active processes on a system with protected memory. Since memory is protected, normally only the privileged user *root* could scan memory to list these processes. However, this would prevent other users from keeping track of their jobs. As with the history file, the solution is to use a setuid program, with *root* privileges, to read memory and list the active processes.

Setuid programs introduce many security problems [TRUS80]; many of these can be dealt with by programming very carefully. The reader should bear in mind that on some systems, the mere existence of a setuid program introduces security holes; however, it is possible to eliminate the obvious ones.

By following some simple rules, programmers can decrease the danger of a setuid program being able to compromise system safety. These rules are:

†UNIX is a Trademark of Bell Laboratories.

1. Be as restrictive as possible in choosing the UID and GID of the setuid program.
2. Do not write interpreted setuid or setgid scripts.
3. Do not use *creat*(2) for locking.
4. Catch all signals.
5. Check data for consistency.
6. Take extreme care when recovering from errors.
7. Close all but necessary file descriptors before calling *exec*(2).
8. Reset effective UIDs and GIDs before calling. *exec*(2).
9. Check the environment of the process.
10. Be careful with I/O operations.

Setuid programs explicitly violate the protection scheme designed into UNIX. On systems where security is not a problem, this is a blessing, since it enables many things to be done easily that would otherwise be very difficult; but on systems where security is a problem, these programs also pose very real threats. Unfortunately, they are very necessary; so, the designers and implementors of setuid programs should take great care when writing them.

Acknowledgements: Thanks to Bob Brown, Peter Denning, George Gobel, Chris Kent, Rich Kulawiec, Dawn Maneval, and Kirk Smith, who made many constructive suggestions.

References

[ALEP71] Aleph-Null, "Computer Recreations," *Software — Practise and Experience* 1(2) pp. 201 — 204 (April — June 1971)

[TRUS80] Truscott, Tom and Ellis, James, "On the Correctness of Set-User-ID Programs," Department of Computer Science, Duke University (unpublished)

OPERATING SYSTEMS SESSION II

Raymond Benoît

Environnement Canada
Montrêal, Canada

The second operating systems session consisted of an experience panel, a presentation by Cray Research and a report from the operating systems committee chairman.

In the experience panel part Jim Sherin talked about Westinghouse´s experiences in installing COS 1.14 on their Cray 1-S and on the Cray X-MP 48 they are managing for the Pittsburg NSF site. Mostyn Lewis then shared the experience he acquired during the installation of COS 1.15 and the testing out of the permanent file archiving package on Chevron´s machine. Kent Koeninger, from Apple Computer Inc., also reported his findings on the first installation of UNICOS in native mode on their brand new X-MP 48.

Tom Lanzatella, from Cray Research Inc., gave a presentation on FSS pre-emption, a COS feature that will be released soon and will help manage Fast Secondary Storage devices such as Buffer Memory and SSD. In the last part of the session I gave a status report on the committee´s activities, this report is integrated with the final committee´s report found at the end of these proceedings.

I would like to thank all the speakers for the fine job they did in presenting their papers and for participating in this OSC session.

COS 1.14 Experiences at Westinghouse from
June 1985 to May 1986.

Over the past 11 months at Westinghouse, our software team has installed COS 1.14 on several different mainframes. We feel that our experiences may be of use to those who have not done this and bring back memories (both good and bad) for those who have already done this.

As a review Westinghouse installed its first Cray-1S in June 1981. This Cray was without an IOP and ran COS 1.10/CFT 1.10(prerelease). In February 1983, Westinghouse went to COS 1.11/CFT 1.10 and installed IOP SN 11 with STC on-line tape drives. In April 1984 SN40/11 CRAY-1S was installed at Westinghouse for COS 1.12/CFT 1.11 use. COS 1.12/CFT 1.11 was installed into production use by the end of February 1985.

As we as a software group have to provide to our users equal or greater functionality whenever we upgrade operating systems. Because of this we've made special efforts to drag along older versions of the Cray compilers and libraries for those codes who have 'teething' problems with the newer Cray versions.

The conversion of our users to COS 1.12 was especially painful as some changes had to be made to library routines because of the introduction of the Time-Stamp-Unit concept. Since many of our codes are Configuration controlled because of their use in Nuclear work, this meant a lot of paper work for our users to justify reloading their codes and many hours of reverifying their work so that it would be acceptable to the NRC.

So far we do not have COS 1.14 in production use on our CRAY-1S' because of the need to provide for the older compilers and utilities and a period of time for our users to convert to the newer compilers and test out their codes. Our projected schedule for this is May 27, 1986. COS 1.14/CFT 1.14 is now in production use at the NSF site in Pittsburgh managed by Westinghouse for the Mellon-Carnegie-Pitt Corporation as ' The Pittsburgh Super Computing Center'. Getting this software ready was an adventure in itself, which will be detailed below.

In the process of preparing COS 1,14 for the Cray-1S' the following problem areas were discoverd and overcome:

1) The base line of COS 1.14 was changed from a Cray-1s without an IOS to a Cray XMP-1 with an IOS.
2) Security was automatically invoked.
3) The default values of other installation parameters were also changed.
4) Evidentally Cray must assume that everyone runs with the default loader setting core to zero. Because of NRC rules our default loader is set to indefinite so as to catch those people who don't bother to initialize their variables. If zero is what Cray expects to be used to build their products, then it should be placed on every LDR statement in their

build procedures.

5) Once we got a useful version of COS 1.14 ready for user testing we found that we had some problems running codes which used a library routine called SYSINFO. It turned out that in COS 1.14 the Data Name Table (DNT) increased in size to D'24 words from D'20 words. In order not to affect already running absolutes, a kludge had to be concocted to pass back to the use only D'20 whenever the Dataset Name happened to be $OUT. Also, the start of the DNT chain in the JTA was moved which caused some headaches. A local mod was necessary to make an EXP call so that the DNT start could be identified.
6) Once around this hurdle we found that users running CFT 1.11 absolutes with FORTRAN 77 OPENs had problems in aborting with operand range errors in F$OPN inside EXP. Analysis of this problem revealed that if the second word of the Open Dataset Name entry was non-zero, then this value was assumed to be the pointer to the DNT entry. Again this was caused by our need to initiate core to an indefinite value. After two attempts to resolve this dilemma, the solution was to test the second word for indefinite and if it was then to clear the word to zero so that the EXP code could proceed as if nothing had happened.

7) The next adventure occurred when CFT 1.10 absolutes were tried under COS 1.14. The error that occurred was that thejobs aborted with the arcane message 'Invalid Dataset Name'. Investigation of this problem revealed that at COS whatever EXP function code O'11 was the call to the EXP function F$MEM. At COS 1.14 this function was now a call to F$QIO. The fix for this was relatively simple: namely to let it fail the test for file name and jump to an area where the call was transferred to F$MEM in COS 1.14. This code got us around the original problem, but another one was now revealed. This error occurred in $TMGR in $SYSLIB. A request was being made for the maximum memory available in the machine. This was not being returned to the user at all. This caused a lot of calculations to become negative and hence stop program execution. Once again code could not be inserted in the loader, because our current CFT 1.10 absolutes had to run. Once again a kludge was put into EXP to trap out this call and return the maximum available memory to the users. This has caused our CFT 1.10 absolutes to execute in a proper fashion.

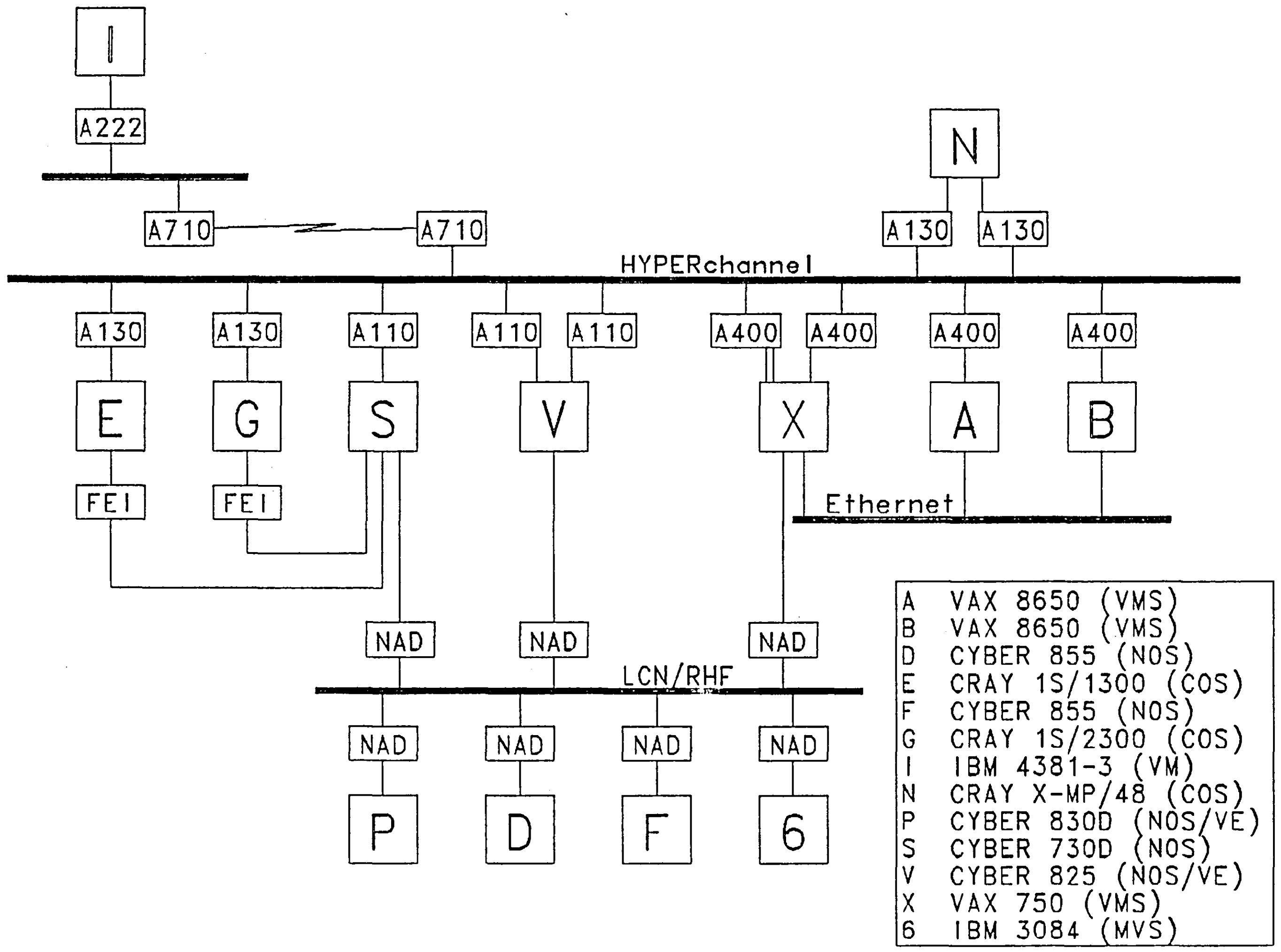
8) We cannot read COS 1.14 written tapes on a COS 1.12 system due to a label problem. When attempts are made to read the tape, the message 'WRONG LABEL TYPE (FRN)' appears and the job must be aborted. TQM was rewritten for COS 1.14 because of the new features to be provided.
9) After we had converted our local mods over to COS 1.14, it was thought that configuring an XMP would be a simple exercise. This was not to be. The sample XMP-48 configuration from Mendota was used to put together a configuration for the NSF machine prior to our early March 1986 trip to test out SN 211/105 in Chippewa Falls. After floundering for two days trying to get this to successfully work, this effort was abandoned in favor of using the methodology used by the folks in Chippewa to generate systems for many differ-

ent Cray systems. This allowed a useful system to be generated for
the Pittsburgh install. The effort to decode what was
necessary for the IOS configuration of VAX stations consumed
several more days. A completely useful system is now in use on the
the NSF machine.

As one can tell we've had an exciting time at Westinghouse with COS/CFT 1.14. As a point of interest, the following attachments are made to this report:

1) Attachment A details the Westinghouse Engineering Local Area Network. Work has commenced to link all the CRAYs to the RHF network as co-equal nodes withe the NOS machines.
2) Attachment B details the configurations of the CRAY-1S' in the ESCC (Energy Systems Computer Center).
3) Attachment C details the configuration of the XMP-48 installed in the Pittsburh Supercomputing Center.

ATTACHMENT A

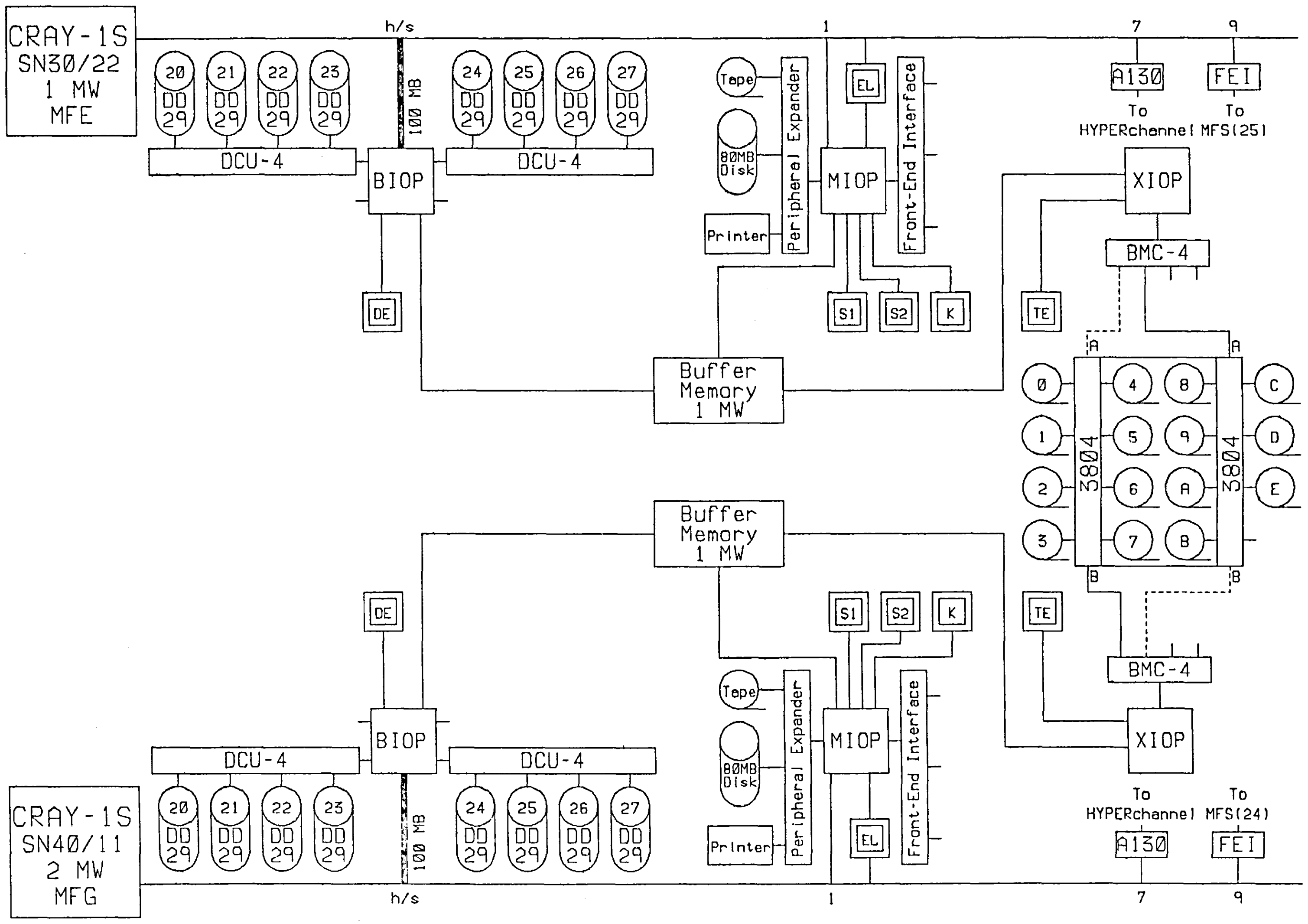


A	VAX 8650 (VMS)
B	VAX 8650 (VMS)
D	CYBER 855 (NOS)
E	CRAY 1S/1300 (COS)
F	CYBER 855 (NOS)
G	CRAY 1S/2300 (COS)
I	IBM 4381-3 (VM)
N	CRAY X-MP/48 (COS)
P	CYBER 830D (NOS/VE)
S	CYBER 730D (NOS)
V	CYBER 825 (NOS/VE)
X	VAX 750 (VMS)
6	IBM 3084 (MVS)

Engineering Computer Local Area Network

04/17/88 RRP

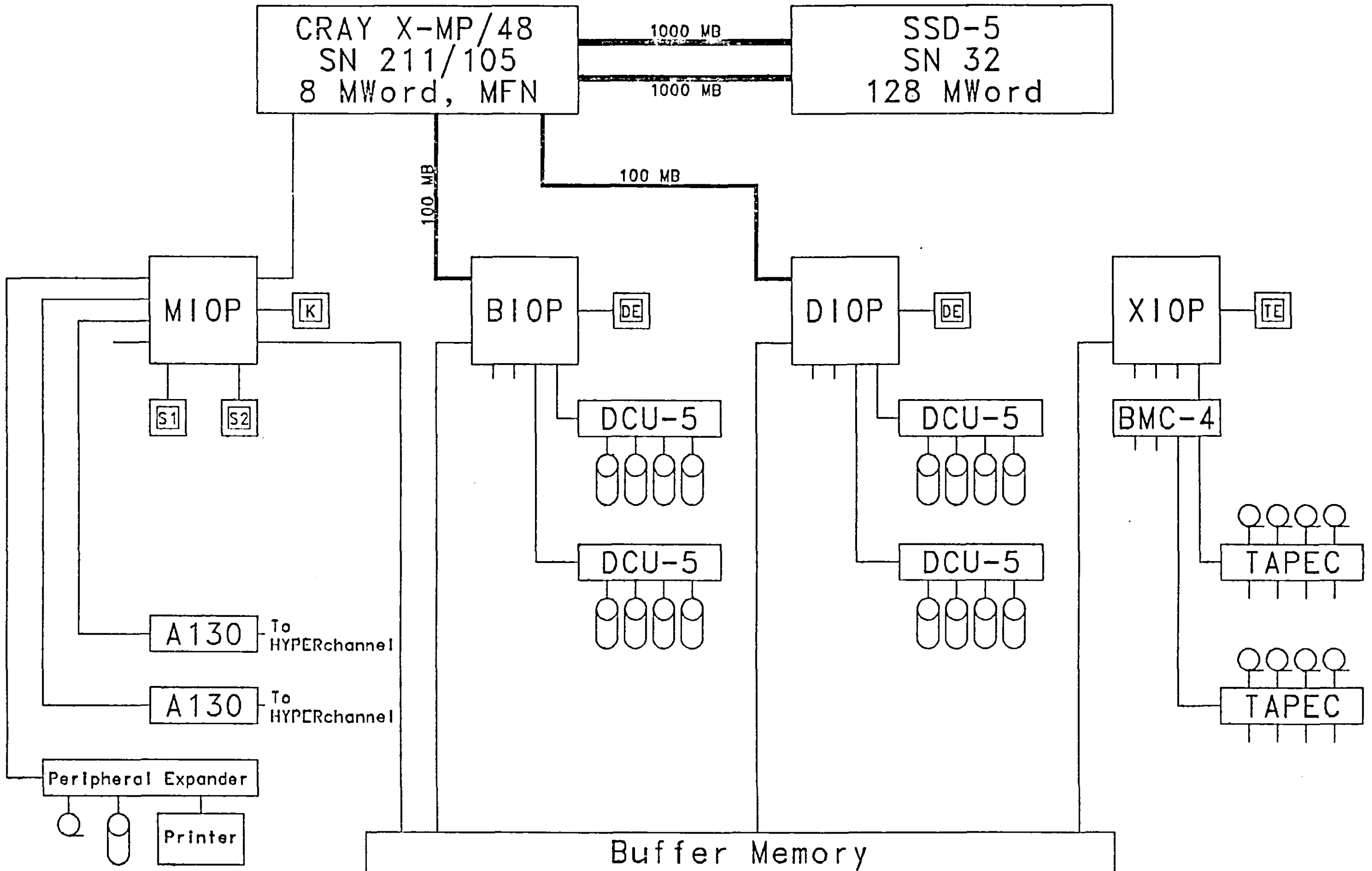
ATTACHMENT 13



CRAY Channel Configuration

02/12/86

Attachment C



Buffer Memory
4 MWord

Pittsburgh Supercomputing Center
CRAY Configuration

02/20/86

COS 1.15 EXPERIENCE

Mostyn Lewis
Chevron Oil Field Research Company
La Habra, California

This experience covers a "pre-release" of COS 1.15 (actually three versions designated D, M, and Q) used over two months. It was run on an X-MP/48 using four DD-49 disks exclusively reserved for 1.15 testing.

First, we found we could not make PASCAL without having the 1.14 BF4 PASCAL. (We were running a 1.14 BF1.) PASCAL was necessary as part of $UTLIB is written in it. Also, PASCAL had a peculiarity where it internally checked its SEGRES consistency with the system's SEGLDR. So, to get around all this, we first made $UTLIB without the PASCAL content and then built SEGLDR. Next, PASCAL was built with an appropriate PASCAL and $UTLIB redone with the new PASCAL and with its PASCAL content intact. Then we went around again for safety.

Our first 1.15 crash was a reoccurrence of an old bug. We had buffer memory as the first entry in the EQT and startup crashes when it tries to use it for parameter file work.

Our major purposes in testing COS 1.15 were to be ready and familiarize ourselves with Space Management/Dataset Archiving and GOS (for UNICOS testing). COS 1.15 has a plethora of catalogs, the $MCD, $DSC, $BCD, $BVCD, and the $DXT. First time you need to create catalogs with GENCAT. The documentation was erroneous in its use. Also, you can NOT recover the $MCD as advertised and have to re-create it every restart. There were hassles with the $BVCD (which lists tape volumes). You need to tell it the length of a tape in megawords but the manual said megabytes -- imagine the consequences!

The backup job (master job) needs to access all the datasets it is going to backup in UQ mode at the same time -- this can need gigantic PDS sizes (which now have two word entries). Datasets are sorted by size so they fit tape volumes efficiently (if you get the length of the tape right). Jobs are spun off but only two at a time (system job limits).

We found various essential parameters undocumented in manuals but present in the code. For example, $ADMLIB (from UTILPL) contains a routine called spintape. Spintape shoots off JCL with JOB and ACCOUNT cards which have been modified at many sites. We had to modify the code to suit our own operating parameters. The backup jobs call two verbs, BUPIO and RECIO, which needed to be in the system directory but were not (again, no documentation).

Then, the $BCD catalog (defined to be 200 sectors long) filled up with 800 dataset entries. It was increased to 1000 sectors. Really, we expected some meaningful parameter, such as number of files, not size in sectors! We were disappointed in the lack (deferred implementation) of front-end migration because of the excellent match this would have made with IBM's TMS and/or HSM.

It was necessary to always exclude certain datasets from backup such as $VALIDATION, $ACCOUNT, and $BULLETIN (again, not in any documentation) as UQ mode on any of these stops the system. All catalogs need to be omitted as well. (Also, leave out the SDR entries if you don't like error messages.)

Whatever you do, don't drop any GENCAT/BACKUP/MANAGE/RECALL jobs or their spawn, else you will suffer a PDM hang. There are warnings in the manual on this but is a dangerous operational aspect.

Only non-accessed datasets can be backed up. Hence, beware the wrath of a user whose dataset was not backed up because he was using it. (Long-running jobs are obviously prone to this.)

There is a danger in assigning very large chunks of disk space (by mistake or by design) because if not enough space is left, many migrated datasets will be deleted until there is enough space to satisfy the assign, leaving other users and not to mention a disgruntled Operations staff to poor turnaround while their datasets are reloaded off tape.

Off the subject of datasets, we noticed excessive wait for memory states when the machine was not very busy; jobs seemed to be ignored by the job scheduler. We had trouble with a new security privilege, SCWNSC, which got in the way of random use of disk (not writing sequentially). We had to remove checking for this from EXP.

Tapes seemed to work fine with auto volume recognition and multi-volume along as they were transparent. However, our long-standing grief with VBS tapes returned. We could not accomplish EOV processing and jobs aborted with uncleared DSP errors. We received various mods but none that fixed the problems.

The MVS station showed up a degradation in interactive response -- under 1.14 we could get about half a screen of data at a time back from interactive but under 1.15, we got a "klunky" one line at a time!

It should be pointed out that all the above was with pre-release software and we expect Cray to fix it. However, there is no better test or feedback than a willing user for future software. We hope Cray continues to allow pre-releases to sites so there can be mutual benefits. Beta testing is generally too restrictive and the more people who can try a system before it is released, the better.

A Report on the First Native X-MP Unicos Installation

R. Kent Koeninger
Cray Evangelist
Apple Computer, Inc.
Cupertino, California

Why a Cray for Apple?

Apple Computer, Inc. purchased a Cray X-MP/48 to shorten the development time necessary to get a new product to market. The Cray will be used to simulate the "desktop look and feel" of Apple products early in their development cycle, before the product is physically prototyped. It will also be used for VLSI and logic simulations.

With the X-MP, developing new revolutionary Apple products will be easier. It will provide Apple engineers with state-of-the art tools.

Operating Systems and Languages

Apple chose the Unicos operating system because our most important applications are interactive and Unicos was in the process of being released. "C" is the dominant language on Apple's Cray because this is the development language in use by the software engineers on the project. Both Unicos and Cray-C are very young and have much room for improvement.

Graphics

Apple's Cray will have a graphics frame buffer attached to the HSX channel on the IOS. This will be a high resolution, dense color, rapidly refreshed, graphics display.

The graphics resolution will exceed the human eye's ability to resolve pixels. The display will have variable resolution up to 1280 by 1024 pixels, with 24 bits of color. It will be updated at least 20 times a second and will be refreshed 60 times a second. The HSX channel will provide 850 megabits/second bandwidth for refreshing the screen. The graphics will be computed in real time on the Cray.

Security

The new revolutionary Apple products require higher computer security than has been traditional at Apple. With Unix it is difficult to prevent unauthorized access from determined hackers. Unauthorized access will be prevented with physical security; Phones connections will not be allowed to the network on which the Cray is a node. With this physical security, only engineers on the Apple campus will have access to the Cray.

The operating system security will be improved over time.

Installation

The Cray X-MP/48 with Unicos running native was installed on March 26, 1986, three months after it was ordered, and one month after

Apple decided to use Unicos. Cray provided sufficient support to get Unicos up in a relatively short time. The machine was up 98.6% of the acceptance period. The Cray is not yet stressed. Apple is preparing new codes, preparing hardware, and preparing communication networks for the Cray.

Unicos Performance

Cray has done and excellent job in maintaining the flavor of Unix. It looks, smells and tastes like Unix.

One of the few statistics available now is an unoptimized dhrystone benchmark of the C compiler performing 9259 dhrystones/second (6.5 faster than the same benchmark on a VAX 750).

With moderate use, no performance problems have arisen that Cray has not solved quickly. Release 2.0 of Unicos promises to be performance oriented.

Pending Issues

No HYPERchannel driver was available for Ultrix on the VAX 785. Cray is not responsible for other Unix network connections, and DEC has not yet released their HYPERchannel driver for Ultrix. Apple had a BSD 4.2 based HYPERchannel driver at NASA Ames ported to Ultrix. It is working well.

Unicos, release 1.0, does not support single character I/O, preventing users from running visual editors on the Cray. The development of new codes would proceed faster if visual editors were available on the X-MP. Cray has short and long term solutions for the single character feature.

Unicos C compiles very slowly and produces unoptimized code. We recently discovered that compiling one megabyte of C code on Unicos 1.0 takes over 10 minutes. Cray expects to make major improvements in this area soon, reducing the compile time and generating vectorized and optimized code.

The "C" compiler supports only eight character variable names. Most of the codes Apple is porting to the Cray use names that require uniqueness beyond eight characters. Apple is developing a preprocessor to make long tokens unique in the first eight characters, and Cray is working on expanding C, CAL, and SEGLDR, to use 32 character names.

No symbolic debuggers are currently available for C under Unicos.

Macintosh Distributed Editor and Engineering Workstation

Apple has a cooperative effort with universities in progress to produce a workstation environment that runs on a standard Macintosh and interacts with Unix. The workstation will provide distributed editing and network communications from the Mac to the Cray via AppleTalk, Ethernet, and HYPERchannel. The workstation will use the TCP/IP protocol.

Conclusion

The Cray X-MP/48 runs Unicos well and will be an excellent tool for engineers at Apple to produce new revolutionary products.

Design for FSS Preemption

Thomas W. Lanzatella

Development Analyst, COS Features Group, Cray Research, Inc., Mendota Heights, Minnesota

Introduction

The term Fast Secondary Storage (FSS) refers to either Solid State Disk (SSD) or Buffer Memory (EBM). These devices are characterized by extremely high transfer rates and are useful to I/O bound applications in the Cray environment. Since FSS devices were originally made available, COS has been enhanced to treat them in such a way that a user job can obtain guaranteed access. The amount of space needed is specified on the JOB command and the job is allowed to initiate when the space is available. The term generic resource is used to describe a device or group of devices whose access is regulated by the system in order to provide guaranteed access or to protect against deadlock.

FSS devices are treated as disk devices by COS. They are accessed through customary use of the COS I/O routines. Recently, COS was changed to distinguish between disk and FSS for charging and statistics gathering purposes. Although the means of access has not changed, usage information related to FSS devices, whether configured as generic resources or not, is now accrued separately from disk usage information.

As the utility of FSS devices increases, their popularity among the user community also increases. Sites with modestly sized FSS devices are faced with the administrative problem of controlling who accesses the device, for how long and at what priority. Since early in 1983, various sites have requested a mechanism in COS which relieves the administrative burden associated with managing FSS devices. The extent of the desired mechanism varies from site to site. The basic need reflected by all sites is an ability to suspend an executing job in such a way that the FSS in use by the job is freed and made available for some other job. Beyond the basic capability, sites would like to have the operating system automatically enforce priority access to an FSS device in the same manner that central memory is shared among all executing jobs.

This design proposes a mechanism, system changes and publications changes needed to support automatically preemptable FSS devices.

Requirements

1. All FSS devices configured must be (optionally) preemptable.
2. The site must be able to specify the degree to which FSS devices are oversubscribed. The operating system refrains from initiating jobs which cause the amount of disk needed to store preempted dataset images from exceeding the amount implied by the degree of oversubscription.
3. The site must be able to specify which disk devices are preferred for holding dataset images swapped from FSS devices.

4. The site must be able to control the preemption process on a global basis or for a specific job.
5. The operator must have access to displays which tell how FSS is being used.
6. Preemption frequency must be a tunable option.
7. A site should always (within reason) be able to initiate a high priority job needing FSS.
8. System performance statistics related to the amount of I/O time spent swapping and the number of swaps are available to the site analyst.

Design Overview

This section illustrates the preemption capability in a broad nonspecific way with emphasis on the design components and how they fit together. The ability to preempt FSS devices in COS is divided into several parts.

1. Organization of disk space for preempted datasets (swap space).
2. Enforcement of oversubscription and job initiation
3. FSS space contention algorithm
4. Data transfer
5. Operator commands and displays
6. Job recovery

Swap space is organized in a way which satisfies requirements 2, 3 and 7. The startup parameter file is the place where a generic resource is declared preemptable, the extent of oversubscription, swap frequency, preferred devices and the way that swap space is partitioned.

Swap space is the volume of disk which the site needs to accommodate an oversubscription factor. Swap space is not isolated from use by the rest of the system unless preferred devices which are request-by-name are included in the swap space declaration directives. Users must be urged not to use these devices to achieve isolation. The system does nothing to guarantee that the volume of swap space implied by the oversubscription factor is available.

The operator is informed continuously through the SWAP display as to the availability of swap space. If local and permanent datasets grow so that swap space is not available, a mandatory response message is sent to the operator at five minute intervals and a warning is placed on the SWAP display. Meanwhile, jobs needing preemptable generic resources are not allowed to initiate until the full volume of swap space is available. Jobs in execution continue to share device residency as long as disk space is available for preempted datasets. If disk space is low enough that preemptable generic resources cannot be cleared of datasets, a warning is placed on the SWAP display and preemption is suspended until space is available.

Swap space partitioning is an attribute of swap space which is important in the following situation. Disk is the system resource used to store FSS datasets which are waiting their turn to reside on the FSS device. Due to the size of FSS devices, disk is a limited resource which can be consumed by a relatively few number of FSS images. The extent of oversubscription is therefore a constraining factor for the site analyst. A prominent need among the user community is the ability to always be able to introduce a high priority job needing FSS, to have the device freed by the operating system and to have the job execute. Since the amount of swap space is itself limited, a large number of medium and low priority jobs could consume the available swap space leaving the site unable to deliver prompt service to a high priority job. For this reason, an ability to partition swap space is included in this design. The analyst has the ability to associate a certain percentage of swap space to jobs of a certain priority range.

The following rule is used when allocating swap space. When a job needing resources enters the system, swap space is allocated from the partition associated with the priority of the job. The volume of swap space allocated is equal to the number of sectors declared as the job limit on the JOB command. If the partition is full, space is sought in partitions of lower priority. If space cannot be found in a partition of the same or lower priority, the job does not initiate. Space is never allocated from partitions with a higher priority than the incomming job. This scheme has a minimal impact on the total amount of swap space required and still allows the site a means of delivering prompt service to high priority jobs.

The priority of a job can be changed by the operator. When a job needs preemptable resources, a priority change causes the system to look in a different partition for swap space. When a job is executing and already has a swap space allocation, a priority change can cause the allocation to violate the partitioning rules, particularly if a job's priority is reduced. On this occasion, the system will attempt to realign the swap space allocation so that it adheres to the partitioning rules. However, this may not always be possible since a lower priority partition may be full. When a realignment cannot be performed, an indicator is set on the RSTAT display and the job continues executing. Jobs needing space from the partition occupied by the job must wait for the realignment to occur or for the job to terminate.

The operator has the ability to remove a preemptable resource from use (requirement 4). A new operator command, SWEEP, is installed for this purpose. When a resource is removed from use, the resident datasets are removed and stored in swap space. Swap space must be sufficient to accommodate a sweep resource request from the operator. An oversubscription factor of one reserves a volume of swap space equal to one image of the resource. The resource can then be swept but it cannot be oversubscribed. An oversubscription factor of two reserves a volume of space equal to two images of the resource allowing an oversubscription ratio of 2:1.

This aspect of the design conflicts with an existing mechanism in COS, flush volatile device. The flush device mechanism makes a literal copy of a volatile device on a preallocated area of disk. The flush mechanism is useful when an FSS device is not configured as a generic resource or as a nonpreemptable generic resource. The flush mechanism must be left intact. Sites choosing preemption need a method of disabling the redundant flush mechanism. An option is therefore included among the STARTUP parameter directives which turns off the reservation of a flush area.

The SWEEP command has three forms. The first allows all preemptable generic resources to be emptied and left ineligible for use. The second form is identical to the first except it is specific to a single resource. The third form causes datasets associated with an individual job residing on any preemptable resource to be moved to swap space and the job to be operator suspended. If a job which has been individually swept by the operator is resumed, the operator suspend state is rescinded and the job becomes eligible to use the device at its current priority.

A new operator command, RESTORE, is installed which rescinds the effects of the SWEEP command. The first two forms of the command are identical in syntax to the first two forms of the SWEEP command and serve to reinstate resource availability. The third form is specific to an individual job and causes an immediate effort to restore swapped datasets to the appropriate resource. The command does not rescind an operator suspend and has no effect on the priority of the job. If space is not available on the resource needed, jobs are removed in priority order, lowest priority first, until space is available.

Job initiation occurs when swap space is available. The job is allowed to execute up to the point where it makes its first request to open a dataset on any of its preemptable resources. The system decides if enough space to satisfy the jobs requested limit from the JOB command is available. If the device if full, the task associated with the request is placed in a special category of the event-wait state and contends for space.

Space contention is mediated by the job scheduler. The process is governed by three objectives.

1. The highest priority jobs have access to the device before low priority jobs.
2. A job which succeeds in gaining a device allocation is guaranteed residency for a set period of time (thrash lock) or until manual intervention.
3. Jobs of equal priority share the device in a round-robin fashion.

These objectives result in a process where priority access to a device outweighs maximum device utilization. Additionally, due to the requirement that all FSS devices be (optionally) preemptable (requirement 1), situations arise where a high priority job needing access to all preemptable devices can block access to an empty device while waiting for a full device to become available.

These disadvantages are offset by observing that an aggressive effort to maximize device utilization through occupancy would result in a certain number of low priority jobs with device access. Since low priority jobs are not in memory as often as high priority jobs, device utilization would suffer. Regarding the device blockage problem, sites may opt to configure only one preemptable device.

Several events cause the scheduler to seek canditates for device residency.

1. Manual commands, SWEEP and RESTORE, from the operator
2. Initial space requests originating from user tasks
3 Dataset release without HOLD
4. Job termination
5. Thrash lock expiration

The thrash lock is specified among the startup parameter file directives and is universal for all preemptable devices.

The job scheduler manages the data transfer process. When a swap is necessary, a current roll image of the job is produced and the JTA of the job is placed in memory. The DNT chain is scanned for datasets residing on the affected device. When a dataset is found, only the data corresponding to the DAT partition associated with the device is moved. If a new DAT partition must be larger than the old, the new partition is placed in the STPDAT pool. Each DNT in the chain is examined and processed in this way. When the swap is complete, the new JTA is written back to the roll image. The job is then rolled-in so that DAT's left in the STPDAT pool can be moved into the JTA. Although individual datasets belonging to a job are transferred sequentially, several jobs can be swapped in parallel. The process of producing a roll image before and after a dataset swap ensures job recoverability.

Because track sizes vary among the types of disks and FSS devices which can be connected to the CRAY, a problem is apparent in the data transfer mechanism. The difficulty is that the track size of a disk device is not necessarily the same as that of an FSS device. Consequently, if only the partition associated with the FSS device is copied, the last track of data on the destination device will not be full. The representation of a partial track within the DAT partition is straight-forward but requires minor changes in STARTUP and the disk queue manager (DQM). In order that datasets generated with partial tracks can be accomodated on a predecessor system without the FSS Preemption feature, a bridging mod is required. A bridging mod will be installed for this purpose in COS 1.15.

The RSTAT display is altered in several ways to provide information about preemptable resources. The RSTAT P display indicates total swap space committed for each preemptable resource as well as the resource status (SWEEP/RESTORE). The RSTAT GN grn display indicates swap space available, swap space allocated, resource status, flush space, current oversubscription ratio and thrash lock value. Additionally, each job listed on the display has a status field describing device residency and swap space alignment. The changes made to the station segment used to generate the RSTAT display constitute a protocol change.

Graphics Session

H. E. Kulsrud

IDA/CRD
Princeton, New Jersey

The Graphics Session consisted of four presentations. The first two papers covered the state of the art in graphics as used for scientific computation. "3D Imaging and High Speed DMA Graphics" by Robert W. Conley of AFWL and "Vectorization of Graphics" by Gerald Edgar of BCS are described in detail by their authors.

Robert L. Judd, of LANL described the need for ultra high speed graphics and plans for obtaining it. LANL users have been hampered by slow response times in their use of computer graphics. Much higher data rates are needed to satisfy the graphics requirements of the supercomputer user. This need presents several problems to the computer scientist that involve the design of operating systems, networks, graphic devices and other components of the total system. The presentation covered some of the key issues involved in building a system for ultra speed graphics and ideas that are being considered to extend these high data rates into the offices of the users.

The goal for supercomputer graphics was demonstrated by Larry Yaegar of Digital Productions. He showed several short films and videos illustrating what his company can do when interfacing a supercomputer to the best graphic output devices. The high technical quality of these films and the merging of art and computation was very impressive.

3D IMAGING AND HIGH SPEED DMA GRAPHICS

Robert W. Conley, Jr.

Air Force Weapons Laboratory

Kirtland Air Force Base, New Mexico

INTRODUCTION

Several projects were begun at the AFWL in 1981 to provide new computer graphics capability for the research scientists at the laboratory. These coincided with the introduction of CTSS as the operating system for the Cray-1s; the computer itself had been installed early in 1980. The interactive computing environment of CTSS has the potential to support raster graphics display hardware and powerful image rendering programs. In the intervening years raster graphics has emerged as a significant capability for the display of scientific data, but at the cost of severe requirements placed on computing engines, disk storage, communications bandwidth, and software support.

The projects centered around the concept of a color graphics workstation. Two kinds of workstation can be defined. A close proximity, very high speed workstation can be installed near (or on) the computer floor and serviced with direct channel connections to the computing engine. A remotely installed workstation can be located in the office or laboratory and connected to the computer facility with lower speed communications. To date our workstations have been of the second kind.

The function of the workstation is to provide a scientist with shaded color graphics images quickly enough that they are useful in the feedback loop of the research process. To be useful at all the images must be viewable within minutes of their computation; it is preferable that they be viewable within seconds. Key attributes of significant value to the scientist are having the display station placed in his or her work area, and limiting it to single (or few) user access.

It requires little experience with a supercomputer to become aware of its inherent and significant limitations. Although the CPU engine may be among the fastest available, there is a conspicuous lack of productivity enhancing support facilities for code development and information transfer in and out of the supercomputer. This is not a characteristic unique to our site. Surmounting these shortcomings can involve large amounts of money and human resources. Improvements in several areas would ease the task of integrating computer graphics into a supercomputing environment.

COMPUTING ENGINES

Of the several CPU classes available at the AFWL, the Cray was the natural choice to compute raster images. Because of communications limitations, display resolution was selected to be 512 lines by 512 pixels, with 8 to 24 bits per pixel. Work began with two AED 512's (8 bits per pixel); within a year two Raster Technologies Model One/20 frame buffers (24 bits per pixel) were acquired. The displays are supported as DMA peripherals on LSI-11 microcomputers. The microcomputer workstations each have 30 megabytes of online hard disk storage. Typically 200 to 300 single frames can be available on the workstation, with display times averaging a few seconds per frame.

Images are calculated on the Cray at 24 bits per pixel using any spatial resolution up to 2048 by 2048, and written to a device independent META file. The final device driver employs coordinate transformations and dithering, if necessary, to map the spatial and color resolution to fit the device.

The calculation of a 512 by 512 by 24 bit image requires the computation of about 0.75 megabytes of graphics information. Most image calculations can employ bounded extents or scan line coherence to reduce the number of computations, but even then the calculation is quite large. Minutes of computing time on the Cray-1 are typical. It is not uncommon to consume 50 to

100 CPU hours to compute an animation requiring 10 to 100 megabytes of image storage in compressed form. No other computer at the laboratory can approach the productivity of the Cray in this application.

GRAPHICS SOFTWARE

The central site computer graphics support is the AFWL META system, a device independent graphics system introduced in 1975. Limited raster graphics capability (monochrome, six bits per pixel) was added in 1977. Using the FR80 film recorder 18 bits per pixel could be generated with three passes, one pass for each primary color. This was superseded in 1982 with the definition of raster line primitives supporting 24 bits per pixel in any image dimension up to 2048 lines of 2048 pixels.

The META file supports absolute raster lines in which each pixel is represented, and relative raster lines in which only those pixel values which have changed from the previous raster line are represented. The device drivers generate command sequences using absolute raster lines or run length encoded lines as supported by the particular device. In building the META file and in generating the device command sequences, that format is selected which minimizes the number of bytes to represent each raster line. Reduction of the volume of information by a factor of ten is common.

The utility software available for raster graphics includes Movie.BYU, locally developed ray tracing programs, an applications library with a memory resident 512 by 512 virtual raster interface, and the META library raster primitives. These tools support the display of scientific data using raster graphics. Both experimental data and computational models have been used to generate single frame and animated sequences.

COMMUNICATIONS

The fundamental obstacle to be overcome has always been the communications bottleneck; this is no less so today than in 1981. One of the greatest enhancements to a supercomputing environment would be the support of a **standard** hardware and software protocol for high speed information transfer.

The AFWL computers are connected by a 50 megabit per second HyperChannel. With it files may be transferred between mainframes and to and from the mass storage facility. Data paths to offices are TTY connections via Ethernet, twisted pair, or telephone cable running at 9600 baud maximum. It is this bandwith that severely restricts the productivity potential of raster graphics. Even with the reduced number of bytes resulting from data compression, a raster image can take several minutes to transfer.

Several Cray sites have designed and built special purpose hardware and software to increase the transfer rate to and from the computer. This is at best a marginal solution because the level of communications available for use at other sites is not raised. Most sites lack the fabrication facilities to support custom hardware.

Given the low cost coaxial cable standards now emerging (e.g., EtherNet), it is timely to have the mainframe vendor adopt such a standard for DMA speed information transfers. Off the shelf technology can support burst rate transfers in the range of 1 to 10 megabaud. This needs to be as commonly available as TTY hardware.

With the introduction of computers as powerful as the Cray 2, another two orders of magnitude increase in information transfer becomes mandatory. Here too a **standard** -- as opposed to custom built -- I/O connection is needed. Computational models in three spatial dimensions and time (i.e., animation) are now possible as never before. Transfer of an image in less than one second is needed to put animation in the feedback loop of a scientific research project.

A typical animation computed at the AFWL uses 30 megabytes. 9600 baud communications with a mainframe rarely averages more than 200 bytes per second due to system loading and transfer protocol overhead. Days of transfer time are required to move an animation to the workstation for viewing or filming.

LANGUAGE ENVIRONMENT

It is unfortunate that today almost any microcomputer has a better code development environment than any supercomputer. No supercomputer, for example, has the integrated environment support of a high level, algorithmic language that approaches that of Turbo Pascal, a product costing less than $100. The resulting loss of productivity in any research endeavor which depends on supercomputers is enormous. A ray tracing code, for example, centers around an inherently recursive algorithm best suited for implementation in a language such as Pascal.

It is time to bring the supercomputer out of the dark ages. What is needed is an integrated environment for an algorithmic language; Modula-2 would be a good choice. The environment should include such features as a language sensitive editor, interactive access to the execution environment at multiple levels, an incremental compiler, and access to the vector capabilities of the machine. The customary edit-compile-link-run sequence should be abandoned.

As envisioned here, the code developer would enter a session in the editor. Components (i.e., partial programs) could be executed as they are constructed. One should not have to wait until the code is complete to begin execution tests. Source code should be compiled up to the cursor location, so that the press of a key would begin execution at once.

Multiple windows would highlight the sequence of execution at the source code level all the way down to the machine level. It should be possible to interrupt the execution to reenter the editor at any point, and any execution error should automatically invoke the editor with the cursor at the suspected point of failure. An interrupted execution should allow data structures to be modified and execution resumed. Vector computations should be available explicitly using procedure calls, or implicitly through automatic compiler vectorization.

Because supercomputers typically route interactive traffic through front-end computers, this environment truly involves a new level of distributed processing sophistication combined with high speed data paths. Graphics output could be routed through the same path to appear on a separate display or on a window of the main display. This level of access to a supercomputer requires attention during hardware and software design by the manufacturer to be successful.

CONCLUSIONS

Our experience shows that advancing computational power brings with it an increasing dependence on color raster graphics to visualize the complex scientific phenomena being studied. Raster graphics in turn places new and especially stringent requirements on the communications and language capabilities of the supercomputer. (Graphics is not unique in this regard, as almost all scientific use of supercomputers could benefit from improvements in communications and integrated environments.) The full potential of the supercomputer cannot be approached without serious attention being concentrated on improving its subsidiary components.

Report of the CUG President

H. E. Kulsrud

IDA/CRD
Princeton, New Jersey

Welcome to the seventeenth meeting of the CRAY User Group, Incorporated. During these seventeen meetings, CUG has experienced rapid growth. Though this growth may be due to the sale and installation of supercomputers by CRI, the generally upward trend in the number of attendees seem to closely resemble the price fluctuations of CRI stock on the New York Stock Exchange. Our first meeting was attended by sixteen people and the current meeting in Seattle by approximately three hundred. We now have enrolled 86 member sites in our society. We have collected dues for this year from 61 and I hope that the deliquent members will dispatch their checks to the Treasurer immediately.

Since the last meeting in Montreal, the various Boards and Committees have been extremely busy. The Finance Committee has prepared a budget and the Treasurer claims our bank balance is $16,219. The Board of Directors have prepared a document called the CUG Members' Handbook which contains the By-laws, Policy Statements and Guidelines for the Installation Delegates and Committees working within CUG. The Handbook has been sent to all Installation Delegates and we suggest that Installation Delegates read and circulate this set of documents, and that Installation Representatives request to see the Handbook if they have not yet seen it. We plan to update this document from time to time and will be issuing some additions before the Fall general meeting.

The number of Special Interest Committees (SICs) has been increased to eight. Ken Neves of Boeing will be the Chair of the Applications SIC. This group will also cover Algorithms and Data Bases. We have been trying to organize this SIC for some time and are really pleased that Ken has accepted this position. We hope that many people will join this committee. The list of the other SICs and their current Chairs is as follows:

Communications	SIC	Dean Smith	ARCO
CTSS	SIC	Jerry Melendez	LANL
Graphics	SIC	Ron Levine	NASA/Ames
Operating Systems	SIC	Ray Benoit	EC
Operations	SIC	Gary Jensen	NCAR
Performance/Evaluation	SIC	Mostyn Lewis	CHEVRON
Software Tools	SIC	Mary Zosel	LLNL

All these SICs encourage attendees to join.

In addition to the SIC, we have formed a new entity in CUG called the Mutual Interest Group (MIG). The first of these is the Aerospace MIG. This group has met several times already and has as its President Jack Sherman of Lockheed.

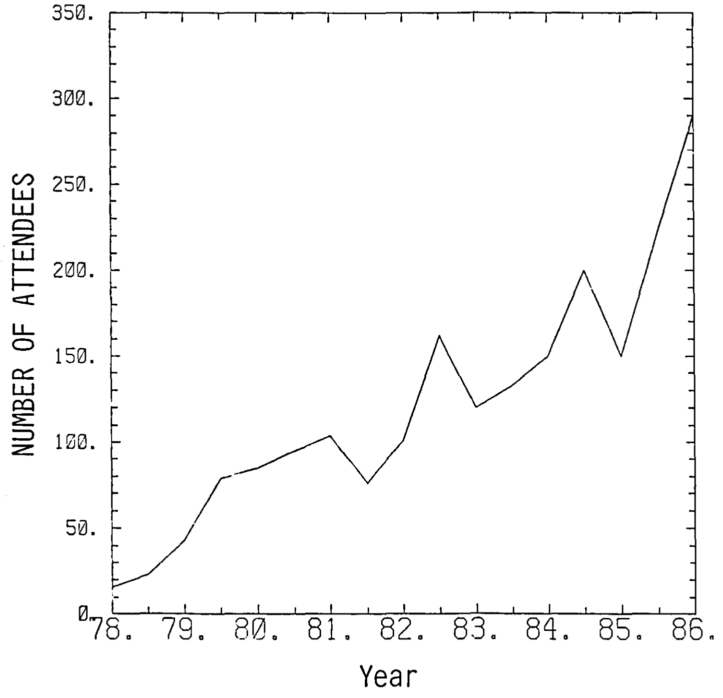
Requests for new and improved software are coordinated by the User Requirements Committee. Steve Niver of Boeing is the Chair and he will be reporting to you on the status of these requests.

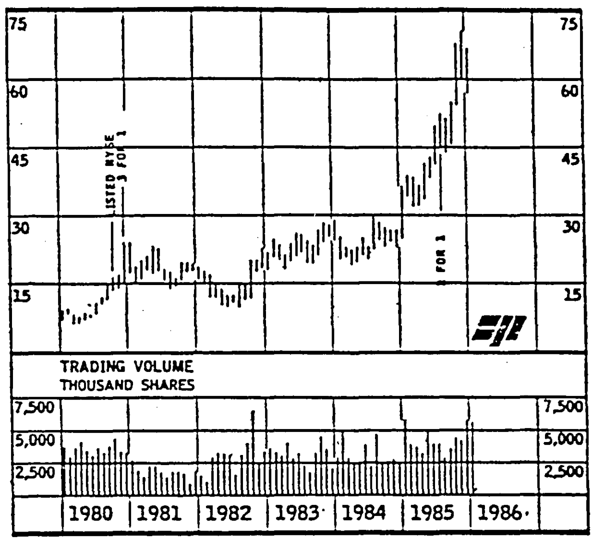
The Program Committee, Chaired by David Lexton of ULCC, was very busy preparing for this conference. The printed program indicates what an excellent job they have done. There are many interesting talks and it is difficult to choose which parallel session to attend. We particularly want to thank Dennis Richie of AT&T Bell for being the keynote speaker of a conference whose theme is UNIX. This committee has begun preparing for the Fall meeting in Garmisch-Partenkirchen, West Germany (September 29 - October 3). DFLVR is the sponsoring site and Peter Herchenbach will be the Local Arrangements Chair.

The Seattle Local Arrangements Committee with Steve Niver of Boeing as its Chair, has been extremely busy. Their arrangements including hotel, meeting facilities and Night Out have been wonderful and we thank them appropriately.

The CUG Board of Directors has accepted the offer of Toshiba and a CUG meeting is now being planned for Toyko in the Fall 1988. We intend to issue registration information well in advance for this meeting. However, anyone interested in attending should begin maneuvering now for their trip to the Orient.

My personal thanks to all the members of all the committees especially the Advisory Council and the Board of Directors who have been working continuously behind the scenes. I also thank the attendees for their enthusiasm which makes a CUG meeting so profitable and enjoyable.





Common Share Earnings ($)

Vice-President's Report

David Lexton

ULCC

The main task of the Vice-President continues to be the preparation of the program and, using the pattern established by my predecessor, Laney Kulsrud, this went quite smoothly for Seattle. All members of the Program Committee, which consists of SIC chairs, local arrangements committee chairs and CRI representatives, attended the meeting of the committee on 7th May. The following items were discussed:

Seattle CUG

The overall organisation, with Steve Niver (Boeing) responsible for local arrangements, was excellent. However, some attendees found the accommodation too expensive although an alternative cheaper hotel had been arranged. The possibility of specifying an all-inclusive conference fee was raised. The format and production of the Proceedings were discussed, the latter because Karen Friedman (NCAR) will no longer be able to do them. Her successor as Proceedings Editor will have a high standard to follow.

Garmisch-Partenkirchen CUG

The theme of the meeting was confirmed to be Applications and Algorithms with the new Application Special Interest Committee under Ken Neves (Boeing) organising some of the parallel sessions. A range of topics was suggested and SIC chairs specified their requirements for parallel sessions.

Planned Future Meetings of CUG

Spring 1987, New York, Grumman. (Reliability of Software and Hardware)

Fall 1987, Bologna, CINECA. (Distributed Computing)

Spring 1988, Minneapolis, University of Minnesota.

Fall 1988, Tokyo, Toshiba.

COMMITTEE REPORTS

Communications (CSIC)

Dean Smith (ARCO)

Deals with issues of networking, station products, protocols and hardware linking Cray computers to other computer systems.

Recognised areas of concern include, and are not limited to:

- Networking management
- Networking
- Superlink
- Hardware requirements
- Station software

I invite all interested parties to involve themselves in the CSIC.

Those people responsible

Members	Observers	Other interested parties
Annabella Deck - Chevron	Eugene Siciunas - Univ. of Toronto	Kurt McMillen - Boeing
Sue Greenburg - Univ. of Illinois	Marco Lanzarini - CINECA	Richard Herndon - Apple
Ron Kerry - General Motors	Robert Benway - REI	Juha Doctor - KFA
Don Mackenzie - Grumman	Sandy Moy - Univ. of Illinois	Kapl Auerbach
Sven Sandin (Co.) - SAAB	Arve Dispen - Univ. of Trondheim	Dieter Raith
Regis Schonneere - CUERAM	Dave Lexton - ULCC	Michel Valin - Environment Canada
Dave Thompson - CRI		Dan Drobnis - San Diego Supercomputer
		Carlos Dangelo - Fairchild

CTSS SIC

Jerry Melendez (LANL)

The CTSS SIC is interested in software written for the CTSS operating system. This includes, but is not limited to, CRI product set converted to run on CTSS. The CTSS SIC will attempt to be a focal point to bring together all the sites using the CTSS operating system.

Activity since the Montreal CUG

The CTSS SIC has not been very active since the last CUG. However, at the Montreal CUG the CTSS community agreed to hold a meeting in San Diego to exchange ideas and software that can be shared among all CTSS sites. The CTSS SIC will participate in future CTSS meetings and will attempt to identify needs which can be satisfied by CRI.

CTSS SIC Members

Jerry Melendez (Chair)	Los Alamos National Laboratory
Hilary Jones (Vice Chair)	Sandia National Laboratory, Livermore
Ken Mortensen	Air Force Weapons Laboratory
Sid Karin	San Diego Supercomputer Center
Jerry Berkman	U.C. Berkeley
Sandy Moy or William Bernhard	University of Illinois
Larry Burdhal	NMFECC
Pat Gray	Lawrence Livermore National Laboratory
Bing Young	CRI

Graphics SIC (GRAPHSIC)

Ron Levine (NASA/Ames)

GRAPHSIC met on May 5, 1986.

Members of SIC present:

Laney Kulsrud (IDA), Richard Schultz (IDA), Fred McLain (SDSC), Bob Conley (AFWL), Gerald Edgar (BCS), Bob Judd (LANL), Carol Hunter (LLNL), Ron Levine (NASA/Ames), John Aldag (CRI).

Not present, but known to count themselves as members:

Gene Miya (NASA/Ames), Kent Koeninger (Apple), Jerry Owens (LLNL), Mostyn Lewis (Chevron OFRC). Mostyn will find an alternate representative from his site.

Others volunteering since this meeting: Jean Shuler (LLNL). Also, representatives of several sites indicated that they wanted to be kept informed for the benefit of other interested people at their sites: CINECA (Bologna), University of Illinois, University of California.

This was the first meeting of this SIC. The initiating chairman Laney Kulsrud passed the reins to the new chairman, Ron Levine. It was expressed by John Aldag that the heavy weighting of the members present by US Government Laboratories is a shortcoming. (The addition of representatives from Chevron and Apple partially ameliorates it). A more serious shortcoming in the membership is the absence of European representation. A first action item for the chairman is to canvass the European sites to recruit some members and, hopefully to find a European deputy chairman. Failing this, the chairman will seek to find a non-European deputy chairman who is fairly certain of being able to go to Garmisch. Bob Judd is provisional deputy chairman.

There was discussion of the functions of the SIC, its relations to CUG and to CRI. The chairman has tried to capture this in the statement of terms of reference of the SIC appended to this report. There was discussion of possible topics for the graphics session at the next CUG and for agenda items for the next meeting of the SIC. Topics under consideration are: Graphics Tutorial, graphics standards (including graphical communications standards), the role of hardwired graphics processing in supercomputer installations, and more on high-performance high-bandwidth interactive supercomputer graphics.

The main agenda item at this meeting was a presentation by John Aldag and discussion of a paper entitled "Requirements, Status and Plans for Graphics on Cray Computers". This paper has been distributed to the members about a week before the meeting. It is evidently part of a larger document which is a sort of CRI working paper on user requirements. Some material contained in the paper as subset of the larger document are deemed proprietary and were deleted from the version give to the SIC.

The paper usefully partitions the graphics requirements into three bins by the parameter "bandwidth per graphics device". Probably, the majority of the user sites are interested mostly in the "Traditional Bandwidth", and "High Bandwidth" domains. However, the present committee is dominated by the sites planning or pushing for "Ultra-High Bandwidth" systems. Therefore, the Committee was highly interested in one topic discussed by John but not detailed in the paper, viz. the several high and ultra-high bandwidth "experiments" in progress: VME-based interfaces and the definition of the HSX 100 MB/s channel spec and its release to a few selected vendors and customers.

The report was generally well received by the SIC as a positive step, but there was a considerable criticism in details, suggestions for making it more complete, and complaints about deletion of material. A second meeting was held in the afternoon at which a substantial subcommittee (more than half the people) reviewed the paper page by page. From notes of this meeting the chairman will prepare a short written response of the SIC to the paper, and will forward it to CRI, after having first circulated it to the members of the SIC.

The chairman has taken the liberty of coining the acronym GRAPHSIC, which is also an anagram, if not a state. He will happily withdraw the suggestion if it causes a public outcry.

TERMS OF REFERENCE OF GRAPHSIC

The Special Interest Commitee on Graphics is to serve as a resource for communication among users sharing interests in computer graphics associated with Cray systems and applications. In the relations among its user members, GRAPHSIC expedites the sharing of public domain information about graphics hardware and software systems. In its relation to the rest of CUG, GRAPHSIC has a similar role as an information resource, expanded by a promotional function, with the aim of increasing interest in and awareness of the uses of computer graphics in supercomputer applications. In its relation to CRI, GRAPHSIC is to represent, both through the User Requirements Committee and directly, user needs and priorities for graphics products and support.

Operations SIC (OPSIC)

Gary Jensen (NCAR)

The Seattle sessions of the Operations SIC set a new standard which will be hard to match at future meetings.

If there had been a pre-determined theme for our sessions, it would have been "Reliability and Maintenance". Fran Pellegrino presented the results of his Hardware Reliability Survey in which 52 systems were represented. That is a "fair" response and we hope to do better next time. Gary Shorrel of Cray Research Inc. (CRI) presented Trends in Reliability obtained from his Data Base of all Cray Systems and it must be noted that the results were close to those in our survey. Stewart Drayton (CRI) explained how CRI is dealing with the problem of providing engineers in a period of extreme growth - he also presented a status report on concurrent maintenance and the hardware failure escalation clause. While it was not impressed that we will not have problems in these areas in the future, I am somewhat relieved that they do have plans and goals.

Karen Schaefer of GMR explained her role and that of the Cray in the research function at GM and discussed the working relationship between GMR and EDS, who supplies the labour force of operators.

Don Whiting of CRI explained the differences in the manufacturing process of building Cray X/MP's versus the older systems. He went into detail regarding parts counts, solder joints etc.

Jerry Stirret of BCS presented facts and diagrams that made the tour of the BCS Computer Center easier to comprehend. A walking tour of BCS was not enough for me to understand the full relationship of all that hardware.

The main event of the conference was the back to back discussion of the PM issue from the viewpoint of Marilyn Richards of NMFE, who practises a policy of no PM on a Cray 1A, Cray 1S and an XMP. Stewart Drayton explained his and Cray's feelings regarding the need and benefits of PM. The Chairman assumed the role of referee for these presentations, although no action on his part was required.

That well describes the activities of the sessions of the Operations SIC except for the following information regarding a new committee formed to give aid and support to the SIC.

The new committee is called the "Executive Committee for the Operations SIC" and is chaired, for meetings in America, by Fran Pellegrino of Westinghouse.

We will be asking for at least one representative from each Cray site as well as CRI representatives. All interested parties are invited. The meeting of this Committee will be on the day prior to the start of the Conference, in this case, Monday. It will be approximately 2 hours long and round table format. The Committee will provide the SIC with direction for the meeting program, support the Hardware Reliability Survey and determine the theme for own next meeting.

I feel that there should be more interaction between the site representatives and our program and welcome suggestions that would better ensure that we are covering the topics of most interest.

The operations site representatives will be receiving more information on this, in the mail.

The theme for the meeting in Germany will be "Operating UNICOS". See you there.

Performance and Evaluation SIC (PESIC)

Mostyn R. Lewis (Chevron)

PESIC covers the areas of:

- Peformance
- Optimization
- I/O

Performance includes

- Performance data acquisition and subsequent analysis; this includes workload and throughput analysis, system tuning and modelling, benchmarking and user program profiling.

Optimization includes

- Techniques to improve system performance, general optimization techniques, machine specific optimization and program optimization techniques.

I/O includes

- Methods to improve I/O throughput at the system and user level, I/O techniques in user programs, comparison of I/O methods and I/O benchmarks. User experiience with new peripheral devices such as disks, SSD, tapes and optical devices and exotica and any supporting code. Particularly, pioneering user experience with the latest Cray (or otherwise) peripherals.

The committee is non-partisan with respect to machine type or software.

Since the last CUG we have formed a working group on performance measurement. The group intends to have results, for the next CUG in the fall of 1986, presenting a benchmark comparison between COS 1.15 and UNICOS (Native) and perhaps embracing the GOS environment.

Ann Cowley has resigned the deputy chair and we wish her well in the future. We have volunteers for the deputy chair and will announce the replacement at a future date. We are seeking a European co-chair from KFA, Julich for the next meeting.

Already we have enough speakers for two sessions at the next CUG and have reserved three parallel sessions to accommodate the expected expansion.

Walt Anderson of Cray Research is our Cray representative for COS and related subjects and we are expecting to have a UNICOS contact in the future.

We had 17 attendees at the 'open' meeting and intend to stay 'open' unless we discuss Cray confidential information. There are 14 members and 7 observers identified.

<u>Software Tools (SWSIC)</u>

<u>Margaret Simmons (LANL)</u>

The purpose of the Software Tools Special Interest Committee is to serve as a focal point for technical interest in software tools from Cray Research that make up a programmer's environment. This environment includes languages, compilers, libraries, editors, debuggers, multitasking tools, etc. We also plan to review such tools as well as requirements for new tools, as appropriate. The committee will be both a technical interface to Cray Research and will plan software tools sessions at CUG meetings.

The Montreal CUG meeting was an organisational one for software tools. We held two meetings and sponsored three technical sessions in Seattle. One of our meetings was a technical session where we focused on debuggers and on table formats. We requested and received high-level informative presentations on both subjects from Peter Rigsbee, our CRI member. The second meeting was held to consider requests for modifications to Cray supported language products. Previously, a long list of requested changes was kept by the CUG and dealt with in a manner similar to user requirements. We feel that such a forum is still needed, and therefore, the committee will serve as a "filter" for such requests. Small items will be handled directly by the committee and Cray Research; larger items will be forwarded to the User Requirements Committee. The three technical sessions in Seattle on Software Tools were organized and chaired by Christopher Lazou, Mary Rosel and Elizabeth Williams. We want to thank them for their hard work. The sessions were of a uniform high quality.

We are currently seeking members from unrepresented sites, especially those in Europe and Japan. Anyone who is interested or who knows of someone who is interested should contact Mary Rosel at LLNL or Margaret Simmons at LANL. A current list of members is attached.

<u>Members</u>

Giovanni Erbacci	CINECA
or	
Elda Rossi	
Peter Rigsbee	Cray
Jerry Berkman	UC Berkeley
Laney Kulsrud	IDA
Karen Pischel	Nasa-Lewis
John Barton	Nasa-Ames
Michel Valin	Environment Canada
Chris Lazou	ULCC
M. Simmons	LANL
M. Zosel	LLNL
Peter Nelson	AT & T

OPERATING SYSTEMS COMMITTEE REPORT

Raymond Benoît

Environnement Canada
Montréal, Canada

TERMS OF REFERENCE

The general goal of the Operating Systems Committee or OSC is to favor the exchange of information between users and between users and CRI. Our role is to provide a detailed technical interface with CRI and assist in planning the technical content of CUG general meetings. The committee's areas of interest are numerous, varied and cover both the COS and UNICOS operating systems. They include maintenance and installation of systems (configuration, generation, modification, testing, tuning...etc), data center support (dumps, validation, accounting and resource control, scheduling, conversion...etc) and many others (security, benchmarking...etc). Some overlap with other committees is unavoidable but is kept to a minimum.

The main tools the committee has to meet its objectives are as follows.

- General conference sessions where users and/or CRI can make presentations to all users on subjects of interest to the OSC.

- OSC sponsored sessions or workshops normally held in parallel with other sessions and attended by a smaller number of users.

- OSC closed meetings held during the CUG conference to facilitate planning and detailled discussions with CRI. These meetings are attended by committee members only. Any CUG site can join the committee and attend the closed meetings by making prior arrangements with the committee chair. Normally there should not be more than one permanent member per installation and a minimum level of activity (attendance at meetings, responding to communications...etc) is required of any OSC member.

- All other OSC activities that take place between CUG conferences. These can take the form of design document reviews, mailings, phone calls...etc.

BUSINESS SINCE LAST MEETING

- Supplying papers and reports for last conference proceedings (D.Lexton).

- Reception from CRI of three design documents (one on FSS premption, one on ressource management and one on user exits) and a request for input on UNICOS features.

- Two mailings were sent out to members containing various information and requests for input on user requirements, design documents...etc.

- Planning the current conference sessions (the OSC I and OSC II sessions plus two closed sessions).

CURRENT BUSINESS

The operating systems committee met on Monday the 5th and on Thursday the 8th, no changes were made to the agenda and there were no outstanding action items from the last conference.

Chairman report

I gave a report on the status of the committee. This included a presentation of new and old committee members, a discussion of the new committee guidelines, a discussion of the documents received from Don Mason (CRI) and the two mailings sent out to committee members.

CRI report

Don Mason (CRI) and Jim Miller (CRI) then gave a detailled report on COS and UNICOS. It was suggested that some of this information be given to all users in the OSC parallel sessions to complement the more general overview usually given by M.Loftus (CRI) in the conference opening sessions.

OSC user requirements

The OSC then examined in some detail CRI's response to some operating systems requirements and policy issues stemming from a request by Steve Niver, chair of the User Requirements committee (see report from that committee for details). The majority of the discussions were on the following three main issues:

-Customer access to the SPR database. CRI's responded that it did not plan to provide access to the database by other than Cray employees. The Committee had very strong feelings about this issue and felt it should be pursued further. It was brought up at the User Requirements committee and it will be handled by higher instances of CUG and CRI.

-Memory growth of system utilities. CRI indicated that they were sensitive to the issue and efforts were being made to keep them as small as possible. Menbers of the committee also mentionned that memory increases in system code, in libraries and tables were a problem. Suggestions were made to use more assembly parameters for conditional code (such as ISP library code), to advertise increases in the release bulletins (how much, where and how to reduce it), to also indicate when options are turned on (release,bugfix)...etc. The concept had already been accepted by CRI following a winter 84 ballot requirement? If you have any specific suggestions send them to me.

- Binary compatibility. CRI responded that they did not intend to unnecessarly make changes that render binaries incompatible. The Committee felt that a stronger and more positive committement by CRI was necessary. It is not sufficient to not intentionnaly make incompatible changes, one must intend to make things compatible in the first place if any progress is to be made. Several users have done simple bridging mods in the past to insure this compatibility and CRI could make the same efforts. Several members also complained that some incompatibilities are not advertised in release bulletins or communicated to users even after several sites have encountered them during system upgrades.

Concerning the old requirements on user exits and installation table space a proposed design overview was received from CRI and distributed to OSC members. Conrad Kimball (Boeing) will be coordinating the OSC working group on these issues.

Half a dozen new operating system user requirements were brought up for discussion at the User Requirements committee (check your next ballot). It was agreed that more time was required during the OSC parallel sessions for open discussion of new and old user requirements. It was also agreed with Don Mason that requirements submitted to CRI should be considered as applying to both COS and UNICOS except where obviously unapplicable.

CRI requirements

Two design papers were received from CRI, one on FSS preemption (Fast Secondary Storage) preemption and one on Job Ressource management. Both were bulky and received too late for distribution before the conference. Gary Smith (U of Texas) will be coordinating the working group on FSS and I will take care of the one on Job Ressource management. At the last conference CRI had requested some input on certain UNICOS features, additional input was requested on other features just before this conference. Answers to all the above papers and requests for input will be sent to CRI before the Garmish conference.

Miscelaneous

The Committee discussed the difficulties in reaching a faster and more efficient exchange cycle between CRI and the OSC committee (conferences are held only every six months, some design documents received relate to features that are for all practical purposes already implemented...etc) and suggestions are welcomed.

It was decided that a short committee closed session was necessary towards the end of the conference (Thursday) to finish our business (new requirements , next conference...etc). The Committee also decided to meet without the presence of CRI for part of that meeting. This seemed necessary to discuss more freely certain delicate issues (policy, CRI/CUG relationships...etc).

The idea of splitting up the committee along COS and UNOCOS lines or forming subcommittees was also discussed. This was jugded premature for the time being. Several committee members are interested in both systems and would have difficulties attending both committee meetings. Parallel sessions, workshops and working groups will however be organized along those lines whenever possible.

NEXT CONFERENCE

We are planning three OSC sponsered sessions at the next conference. Two of these will be workshops, one devoted to COS and one devoted to UNICOS, with an experience panel part, a CRI discussion of the latest release and it´s features and a problem discussion/user requirements part. The other session will be left for papers and presentations on COS and UNICOS. Some of the possible topics of interest for those sessions would be (for COS and UNICOS): security, accounting, ressource control, tuning, scheduling...etc. If you want to present a peper at the next conference please contact Dave Lexton (ULCC) or myself. Two closed sessions are also planned for the Garmish meeting, check the Garmish meeting schedule for the exact dates and times. Some of the issues we will pursue with CRI at the closed meetings will be Beta site testing and COS/UNICOS migration.

CONCLUSION

I want to thank all the speakers for the fine job they did in the OSC sessions and all the OSC committee members for their support and contribution in making this a productive and enjoyable meeting.

The current committee members are:

Raymond Benoît, chair (EC,
Dave Lexton, deputy-chair (ULCC),
Don Mason, CRI interface (CRI),
Conrad Kimball (Boeing),
Jim Sherin (Westinghouse),
Mostyn Lewis (Chevron),
Claus Hilberg (ECMWF),
Kent Koeninger (Apple),
Larry Yaeger (Digital),
Charles Slocomb (LANL),
Lothar Wollschlager (KFA).

Members who have joined at this meeting:

Gary Smith (U.of Texas),
Brian Vohs (Exxon),
Sanzio Bassini (CINECA),
Urszula Frydman (U. of California).

Regular contacts, attendees, help...etc:

Jim Miller (CRI),
Walt Anderson (CRI),
Ronald Kerry (GM),
Regis Schoonheere (CISI),
Serge Hardoin (CCVR),
Tony Hackenberg (NASA/Lewis).

ADDITIONAL INFORMATION

CUG Site Contact List
August 14, 1986

Adam Opel AG (OPEL CRAY)
Bahnhofsplatz
Russelsheim
D-6090
Germany

Installation Delegate
T. Zimmerschied 0049-6142-663797

Air Force Global Weather Central (AFGWC)
AFGWC/SD
Offutt AFB, NE 68113

Installation Delegate
Charles Cook (402)294-5884

Technical Contact
Lt. Rand Huso (402)294-4671
AFGWC/SDDN

Operations Contact
Joe Luteran (402)294-2889
AFGWC/CMO

Air Force Weapons Laboratory (AFWL AD)
AFWL/SI
Kirtland AFB, NM 87117-6008

Installation Delegate
Larry Rapagnani (505)844-0441

Technical and Operations Contact
Mike Gleicher (505)844-9964

Apple Computer (APPLE)
20575 Mariani Avenue
M/S #32-E
Cupertino, CA 95014

Installation Delegate and Operations Contact
Richard Herndon (408)973-6278

Technical Contact
Kent Koeninger (408)996-1010

Arabian American Oil Company (ARAMCO)
EXPEC Computer Center
Box 5000
Dhahran 31311, Saudi Arabia

TELEX: 601220 ARAMCO SJ

Installation Delegate
M. Sadlowski

Technical Contact
Alfred Anderson (011)966-3-87-61188
X-2690

Operations Contact
Gene McHargue (011)966-3-874-1945(or 3830)
Box 10356

Arnold Engineering Development Center (AEDC-CCF)
Central Computer Facility
Arnold Air Force Station, TN
37389

<u>Installation Delegate</u>	
Larry Cunningham	(615)454-7263
AEDC MS 100	
<u>Technical Contact</u>	
Wayne Neese	(615)454-4294
AEDS MS 100	

Atlantic-Richfield Oil & Gas Company (ARCO)
2300 Plano Parkway
Plano, TX 75075

TWX 910 861 4320

TELEX 73-2680

Facsimile Transmission DMS 1000(214) 422-3657

<u>Installation Delegate</u>	
Dean Smith	(214)754-6415
PRC - C2292	
<u>Technical Contact</u>	
B.Y. Chin	(214)422-6627
PRC - 2211	
<u>Operations Contact</u>	
Chuck Murphy	(214)422-6612
PRC - 5141	

Atlas Centre (ACUK)
Rutherford Appleton Laboratory
Chilton, Didcot, Oxfordshire
OX 11 OQX
England

TELEX: 83159

Facsimile Transmission: 0235-44-5808

<u>Installation Delegate and Operations Contact</u>	
D.G. House	0235-44-5515
Room F18	
<u>Technical Contact</u>	
T. Daniels	0235-44-5755
Room F12	

Atomic Energy Research Establishment (HARWELL)
Harwell, Oxfordshire
OX11 ORA, England

TELEX 83135 ATOM HA G

<u>Installation Delegate</u>	
A. E. Taylor	0235-24141, x.3053
H 7.12	
<u>Technical Contact</u>	
Don Sadler	0235-24141, x.3227
Bldg. 8.12	
<u>Operations Contact</u>	
Michael Schomberg	0235-24141, x.3263
Bldg. 8.12	

Atomic Weapons Research Establishment (AWRE)
Aldermaston
Reading, RG7 4PR
England

TELEX 848104 or 848105

Installation Delegate	
L. M. Russell	07356-4111, x.6678
Technical Contact	
P. A. Janes	07356-4111, x.4045
Operations Contact	
M.D.P Fasey	07356-4111, x.6491

AT&T Bell Laboratories (ATTBLMH)
600 Mountain Avenue
Murray Hill, NJ 07974

TELEX 13-8650
Facsimile (201)582-2608
(201)582-6934

Installation Delegate and Technical Contact	
Peter Nelson Room 2F-225A	(201)582-6078
Operations Contact	
R.A. (Tony) Shober Room 2F-247	(201)582-2608

Boeing Computer Services Company (BCS)
Post Office Box 24346
Seattle, WA 98124

Installation Delegate	
Stephen Niver MS 7A-23	(206)763-5073
Operations Contact	
Jim Roetter MS 7C-12	(206)763-5510

BP Exploration (BPLONDON)
Moor Lane
London EC2Y 9BU
United Kingdom

Installation Delegate, Technical and Operations Contact	
M.P. Stanyer	(44)1-920-6156

Centre de Calcul EPFL (EPFL)
Batiment du DMA
Ecublens
CH-1015 Lausanne
Switzerland

TELEX: 25 934 EPFV CH

Installation Delegate	
Pierre Santschi	021/47.22.11 011/41/21/47.22.11 (from USA)
Technical and Operations Contact	
Michel Jaunin	011/41/21/47.22.02

Centre de Calcul Vectoriel Pour la Recherche (CCVR)
Ecole Polytechnique
91128 Palaiseau Cedex
France

TELEX: 691596

Installation Delegate Tor Bloch	60 19 41 53
Technical Contact Maurice Benoit	69 41 82 00, x.2534
Operations Contact Paulette Dreyfus	

Centre Informatique de Dorval (CID)
(Environment Canada)
2121 Trans-Canada Highway
Dorval, Quebec
Canada H9P1J3

Installation Delegate and Technical Contact Raymond Benoit	(514)683-9414
Operations Contact Gary Cross	(514)683-8152

Century Research Center Corporation (CRCC)
3, Nihombashi Honcho 3-chome,Chuo-ku
Tokyo, Japan 103

TELEX 252-4362 CRCNET J

Installation Delegate Mike(Mitsuru) Maruyama	(03) 665-9901
Technical Contact Kazuyoshi Fukushima	(03) 665-9901

CERN (CERN)
European Laboratory for Particle Physics
1211 Geneva 23
Switzerland

TELEX: 419000 CER CH

Installation Delegate Jean-Claude Juvet	022-834935

Chevron Geosciences (CHEV-TEX)
2811 Hayes Road
Houston, TX 77082

Installation Delegate and Technical Contact William Kimball Room 1114	(713)596-2520
Operations Contact Juan Cruz Room 3302	(713)596-2523

Chevron Oil Field Research Company (CHEVRON)
3282 Beach Blvd.
La Habra, CA 90631

TELEX: 176967 via San Francisco

Installation Delegate and Technical Contact
Mostyn Lewis (213)694-9235

Operations Contact
John Kunselman (213)694-7029

CIFRAM (CIFRAM)
(CiSi-Framatome)
BP 24
Gif-sur-Yvette
91190
France

TELEX CISIPSC 691 597 F

Installation Delegate
Louis Bosset 69-08-42-03

Technical Contact
Philippe Van Surrell 69-08-67-05

Operations Contact
Regis Schoonheere 69-08-63-19

Commissariat a l'Energie Atomique/CEL-V (CEA-CEL)
BP 27
94190 Villeneuve St. Georges
France

Installation Delegate
Henri Dauty (1)569-96-60, x.6386

Technical Contact
Martine Gigandet (1)569-96-60, x.6184

Operations Contact
Claude Riviere (1)569-96-60, x.6484

Commissariat a L'Energie Atomique/CEV (CEAT)
Centre D'Etudes de Vaujours
Unite de Calcul
BP 7
77181 Courtry
France

Installation Delegate
Bruno Compoint (1) 868-8413

Technical and Operations Contact
Joseph Harrar (1) 868-8688

Compagnie Generale de Geophysique (CGG)
1, Rue Leon Migaux
BP 56
Massy CEDEX
91301
France

TELEX: CGGEC 692442F

Installation Delegate
Claude Guerin (6) 920.84.08

The Computing Centre at The University of Trondheim (RUNIT)
N-7034 Trondheim NTH
Norway

TELEX: 55620 sintf n

<u>Installation Delegate</u>	
Karl Georg Schjetne	47-7-593100
<u>Technical Contact</u>	
Arve Dispen	47-7-592989
<u>Operations Contact</u>	
Kristian Kuikne	47-7-593020

Conoco, Inc. (CONOCO)
1000 South Pine
Ponca City, OK 74603

<u>Installation Delegate and Technical Contact</u>	
Julian Ford	(405)767-3360
394 Park Building	
<u>Operations Contact</u>	
David Mohler	(415)767-2813
394 Park Building	

Consorzio Interuniversitario per la Gestione
Del Centro di Calcolo Elettronico dell'Italia
Nord-Orientale (CINECA)
6/3 Magnanelli
Casalecchio di Reno
40033
Bologna, Italy

<u>Installation Delegate</u>	
Marco Lanzarini	39-51-576541

Cray Research, Inc.
608 2nd Avenue South
Minneapolis, MN 55402

<u>Administrative Contact</u>	
Mary Amiot	(612)333-5889
<u>Technical and Operations Contact</u>	
Dave Sadler	(612)452-6650

David Taylor Naval Ship R&D Center (DTNSRDC)
Bethesda, MD 20084

<u>Installation Delegate and Technical Contact</u>	
Robert Tinker	(301)227-1428
Code 18923	
Bldg. 17, Rm 105B	
<u>Operations Contact</u>	
Gilbert Gray	(301)227-1270
Code 189	

Deutsche Forschungs- und Versuchs-anstalt fur Luft-
und Raumfahrt (DFVLR)
Oberpfaffenhofen
Muncher Strasse 20
8031 Wessling
West Germany

Telephone: (0)8153/281
TELEX: 526401

Installation Delegate	
Peter Herchenbach	(0)8153/28954

Digital Productions (DIGIPROD)
3416 S. La Cienega Blvd.
Los Angeles, CA 90016

Installation Delegate	
Gary Demos	(213)938-1111
Technical Contact	
Larry Yaeger	(213)938-1111
Operations Contact	
Gordon Garb	(213)938-1111

E.I. DuPont de Nemours, Inc. (DUPONT)
Experimental Station
Wilmington, DE
19803

Installation Delegate	
David Filkin	(302)772-3970
Operations Contact	
James Chang Bldg. 320	

Electricite de France (EDF)
1 Avenue du General de Gaulle
A2-004
92140 Clamart
France

TELEX 270 400 F EDFERIM

Installation Delegate	
Yves Souffez	(1) 765 40 18
Technical Contact	
Bertrand Meyer	(1) 765 41 50 or (1) 765 41 05

European Centre for Medium Range (ECMWF)
Weather Forecasts
Shinfield Park
Reading RG2 9AX
Berkshire, England

TELEX 847908

Installation Delegate	
Geerd-R. Hoffmann	44-734-876000, x.340
Technical Contact	
Claus Hilberg	44-734-876000, x.323
Operations Contact	
Eric Walton	44-734-876000

Exxon Co. USA - EDPC (EXXONUSA)
3616 Richmond
Houston, TX 77046

TWX: (713) 965-7310

Installation Delegate	
Michael Beddingfield	(713)966-6134
Technical Contact	
Brian Vohs 107ST	(713)965-7534
Operations Contact	
Don Smith 245 ST	(713)965-7514

Exxon Production Research Company (EPRCO)
P. O. Box 2189
Houston, TX 77001

TELEX: 910-881-5579 (Answer back: USEPRTX HOU)

Installation Delegate	
T.A. Black N-121	(713)965-4203
Technical Contact	
J.E. Chapman N-121	(713)965-4689
Operations Contact	
D.N. Turner N-180A	(713)965-4407

Fairchild (COMUN)
Gate Array Division
1801 McCarthy Blvd.
Milpitas, CA 95035

Installation Delegate	
Carlos Dangelo	(408)942-2587
Technical Contact	
Carlos Dangelo Hassan Nosrati	(408)942-2680
Operations Contact	
Hassan Nosrati	

Ford Motor Company (FORD)
Engineering Computer Center
MD-1, Room 208
PO Box 2053
Dearborn, MI 48121

Installation Delegate	
Neil St. Charles	(313) 845-8493

General Dynamics Corporation (CF)
Data Systems Division
Central Center
PO Box 748
Fort Worth, TX 76101

TELEX: 768231

Installation Delegate and Technical Contact

M.H. Pittman	(817) 777-3102
Mail Zone 1175	

Operations Contact

H.D. Hollingsworth	(817) 777-3238
Mail Zone 2169	

General Motors Research (GM)
General Motors Technical Center
12 Mile and Mound Roads
Warren, MI 48090-9055

Installation Delegate and Operations Contact

Ronald Kerry	(313) 575-3208

Technical Contact

Dean Hammond	(313) 575-3372

Operations Contact

Karen Schaefer	(313) 575-3237
270 R.AN.B.	

Government Communications Headquarters (GCHQ)
Priors Road
Cheltenham, England

Installation Delegate and Technical Contact

Alan Phillips	0242-521491, x.3185
X34	

Operations Contact

R. Medley	0242-521491, x.3185
F/1208	

Grumman Data Systems (GDS)
1111 Stewart Avenue
Bethpage, NY 11714

Installation Delegate and Technical Contact

James Poplawski	(516)575-2934
MS B34-111	

Operations Contact

Steven Hornacek, Jr.	(516)575-4273

Institute for Defense Analyses (IDA)
Communications Research Division
Thanet Road
Princeton, NJ 08540

Installation Delegate and Operations Contact

Robert Cave	(609)924-4600

Technical Contact

Helene Kulsrud	(609)924-4600

Institute for Defense Analyses
Supercomputing Research Center (SRC)
4380 Forbes Blvd.
Lanham, MD 20706

Installation Delegate

Arthur Lazanoff	(301)731-3725

KFA Julich (KFA)
Postfach 1913
5170 Julich 1
West Germany

TELEX: 833556 KFA D

Installation Delegate

Friedel Hossfeld	02461-61-6402

Technical and Operations Contact

L. Wollschlaeger	02461-61-6420

Koninklijke/Shell Exploratie & Produktie Laboratorium (KSEPL)
Volmerlaan 6
2288 GD Rijswijk (Z.H.)
The Netherlands

TELEX KSEPL NL 31527

Installation Delegate and Technical Contact

A.E. Stormer LS-219	070-112741

Operations Contact

A.A.H. Kardol LS-208	070-112601

Konrad Zuse-Zentrum fur Informationstechnik Berlin (BERLIN)
Heilbronnerstrasse 10
D 1000 Berlin 31
West Germany

TELEX: 183798

Installation Delegate

Jurgen Gottschewski	(030)-3032-233

Lawrence Livermore National Laboratory (LLNL)
PO Box 808
Livermore, CA 94550

TWX 910 386 8339 UCLLL LVMR

Installation Delegate

Joseph Requa L-61	(415)422-1100

Technical Contact

Patrick H. Gray L-60	(415)422-4049

Operations Contact

Pierre Du Bois L-67	(415)422-4007

Lockheed Advanced Aeronautics Company (XMP24110)
Dept. 60-40, Unit 50, Plant 2
PO Box 551
Burbank, CA 91520

Installation Delegate and Technical Contact	
Howard Weinberger	(805)257-5725
Operations Contact	
Doug Ford	(805)257-5720

Lockheed Missile and Space Co. (LOCKHEED)
1111 Lockheed Way
Sunnyvale, CA 94086

TELEX: 346409

Installation Delegate	
Jack Sherman	(408)742-8993
Technical Contact	
Doug Telford	(408)742-0948
Operations Contact	
Jerry Roninger	(408)742-5831

Los Alamos National Laboratory (LANL)
P. O. Box 1663
Los Alamos, NM 87545

Installation Delegate	
Charles Slocomb MS B294	(505)667-5243
Technical Contact	
Margaret Simmons MS B265	(505)667-1749
Christopher Barnes Group X-1, MS E531	(505)667-5000
Operations Contact	
Tom Trezona MS 252	(505)667-4890

Max Planck Institute fur Plasmaphysik (MPI)
8046 Garching
Bei Munchen
West Germany

TELEX 05/215 808

Installation Delegate and Technical Contact	
Johann Gassmann	089-3299-340

McDonnell-Douglas Corporation (MDC)
PO Box 516
St. Louis, MO 63166

Facsimile Transmission: (314)233-6149

Installation Delegate
James R. McCoy (314)233-3425
Dept. W512 - 306/3

Technical Contact
James Miget (314)234-3326
W532 - 306/3/395

Operations Contact
F. Brian Hunt (314)233-4900
W270 - 306/2E/290

Merlin Profilers Limited
1 Duke Street
Woking, Surrey
United Kingdom

Installation Delegate
Paul Blundell

Technical Contact
Andy Wright

Mitsubishi Research Institute, Inc. (MIRI)
2-3-6, Otemachi
Chiyoda-ku
Tokyo, Japan 100

TELEX 222-2287 MRI J

Installation Delegate
Nobuhide Hayakawa (03) 270-9211

Technical and Operations Contact
Shuichi Yamagishi (03) 270-9211

Mobil Exploration & Producing Services, Inc. (MEPSI)
PO Box 900
Dallas, TX 75221

Installation Delegate
Beverly Jackson (214)658-4409

MOD (P.E.), RARDE (RARDE)
Fort Halstead
Sevenoaks, Kent, TN14 7BP
England

TELEX: 95267

Installation Delegate and Technical Contact
Bob Youldon 0732-55211, x.3086
Bldg. 511

NASA/Ames Research Center (NAS)
NAS Projects Office
Moffett Field, CA 94035

Installation Delegate
John Barton (415)694-6837
MS 233-1

NASA/Lewis Research Center (NASA/LE)
21000 Brookpark Road
Cleveland, OH 44135

Installation Delegate	
William McNally	(216)433-4000, x.6650
MS 142-2	

National Cancer Institute (FCRF)
Frederick Cancer Research Facility
Advanced Scientific Computing Laboratory
PO Box B
Frederick, MD 21701

Installation Delegate	
Charles Crum	(301)695-2765
Technical Contact	
Jacob Maizel	(301)695-2532
Operations Contact	
Steve Karwoski	(301)695-2775

National Center for Atmospheric Research (NCAR)
P. O. Box 3000
Boulder, CO 80307

TELEX 45694

Installation Delegate	
Bernie O'Lear	(303)497-1268
Technical Contact	
Eugene Schumacher	(303)497-1264
Operations Contact	
Gary Jensen	(303)497-1289

National Magnetic Fusion Energy
Computer Center (NMFECC)
P. O. Box 5509, L-561
Livermore, CA 94550

TELEX 910-386-8339

Installation Delegate	
Hans Bruijnes	(415)422-4012
Technical Contact	
F. David Storch	(415)422-4004
Operations Contact	
Marilyn Richards	(415)422-4397

National Security Agency (NSA)
Ft. George G. Meade, MD 20755

Installation Delegate	
Bruce Steger	(301)688-6275
T335	
Technical Contact	
C. Thomas Myers	(301)688-6275
T335	
Operations Contact	
Maureen McHugh	(301)688-6198
T152	

Naval Research Laboratory (NRL)
4555 Overlook Avenue S.W.
Washington, DC 20375

Installation Delegate
Harvey Brock (202)767-3886

Nissan Motor Company (NISSAN)
Nissan Technical Center
560-2, Okatsukoku
Atsugi, Kanagawa
243-01
Japan

Telex: J47980

Installation Delegate, Technical and Operations Contact
Mizuho Fukuda 0462-47-5523
Engineerng Computer Applications Section No. 1
Product Development Systems Department

Northrop/Advanced Systems Division (NORTHROP)
8900 E. Washington Blvd.
Pico Rivera, CA 90660-3737

Installation Delegate
Kay D. Barnier (213)948-7845

Technical Contact
John Lehmann (213)948-0213

Operations Contact
Paul Lee (213)942-4718

NTT Electrical Communications Laboratories (NTT)
Nippon Telegraph and Telephone Corporation
3-9-11 Midori-cho
Musashino city, Tokyo 180
Japan

Installation Delegate and Technical Contact
Mikio Sasaki (011) 81-0422-59-2261
Information Processing Services Section
Engineering Department

Operations Contact
Hideaki Maeda (011) 81-0422-59-3845
Information Processing Services Section
Engineering Department

ONERA - Calculateur Aeronautique (ONERA)
BP 72
Chatillon Sous Bagneux
92322
France
TELEX: ONERA 260 907F

Installation Delegate
Jean-Pierre Peltier (1) 6571160, x.2094

Technical Contact
Daniel Colin (1) 6571160, x.3098

Operations Contact
Jean Erceau (1) 6571160, x.2465

Phillips Petroleum Company
418 Information Systems Bldg.
Bartlesville, OK 74004

Installation, Technical and Operations Contact
Arvin Todd (918)661-6426

Pittsburgh Supercomputing Center (PITTSCC)
Carnegie-Mellon University
Mellon Institute 409C
4400 Fifth Avenue
Pittsburgh, PA 15231

Installation Delegate
Michael Levine (412)268-4960

Technical Contact
Beverly Clayton (412)268-4960

Operations Contact
Fran Pellegrino (412)374-4281
Westinghouse Electric Corp.
Monroeville Nuclear Center

Rechenzentrum der Universitat Stuttgart (RUS)
Pfaffenwaldring 57
7000 Stuttgart 80
West Germany

TELEX: 07255445

Installation Delegate
Walter Wehinger 0711-685-5391

Rockwell International Information Systems Center (RI)
PO Box 2515
Mail Code SH10
Seal Beach, CA 90740

TELEX: 910-341-6801 (ROCK ISCW SLBH)

Installation Delegate and Technical Contact
Abraham Levine (213)594-2740

Operations Contact
Joe Henderson (213)594-2283

Royal Aircraft Establishment (RAE)
Bldg. P70
Farnborough, Hants
GU14 6TD
England

TELEX: 858134

Installation Delegate
J.M. Taylor (0252)24461, x.3042

Technical Contact
B.E. Taylor (0252)24461, x.3853

Operations Contact
A.J. Chamberlain (0252)24461, x.2375
Bldg. Q108

SAAB-Scania (SAAB)
Aircraft Division
S-58188 Linkoping
Sweden

TELEX: 50040 SAABLGS

Installation Delegate and Technical Contact	
Sven Sandin	4613 182357
Operations Contact	
Stig Logdberg	4613 182371

Sandia National Laboratories (SNLA)
Albuquerque, NM 87185

Installation Delegate	
Melvin Scott Department 2641	(505)844-4075
Technical Contact	
Frank Mason Division 2641	(505)844-2332
Operations Contact	
Kelly Montoya Department 2630	(505)844-1234

Sandia National Laboratories (SNLL)
PO Box 969, East Avenue
Livermore, CA 94550

Installation Delegate and Technical Contact	
Dona Crawford D8235	(415)422-2192
Operations Contact	
M.H. Pendley D8236	(415)422-2965

San Diego Supercomputer Center (SDSC)
GA Technologies
PO Box 85608
San Diego, CA 92138

TELEX: 695065

Installation Delegate and Technical Contact	
Fred McClain	(619)455-4597
Operations Contact	
Dan Drobnis	(619)455-5020 (619)534-5020 (after 7/1/86)

Schlumberger-Doll Research (SCHLUMBE)
Old Quarry Road
PO Box 307
Ridgefield, CT 06877

TELEX: 643359

Installation Delegate	
Bob Snow	(203)431-5527
Technical Contact	
Raymond Kocian	(203)431-5522
Operations Contact	
Josephine Murray	(203)431-5524

Shell Oil Company (SHELLOIL)
PO Box 20709
Houston, TX 77025

TELEX: 71-378-7530 (answer back - Shell MTM HOU)

Installation Delegate
L.J. Kealy (713)795-3320

Technical Contact
B.D. Huff (713)795-3193
Rm. 5B48

Operations Contact
C.D. Smith (713)795-1696
Rm. 1812

Shell U. K. (SHELLUK)
Rowlandsway Wythenshawe
Manchester M22 5SB
United Kingdom

TELEX: 668613

Installation Delegate
David Cheater 061-499-4357

SNEA (ELF)
Rue Jules Ferry
Pau 64000
France

TELEX: Petra 560 804F

Installation Delegate, Technical and Operations Contact
Michel Morin 59-834146

SOHIO
Geophysical Data Center
1 Lincoln Center
5400 LBJ Freeway
Suite 1200-LB 25
Dallas, TX 75240

Installation Delegate, Technical and Operations Contact
Mark Rehrauer (214)960-4336

SVERDRUP Technology, Inc. (SVERDRUP)
Arnold Air Force Station, TN 37389

Installation Delegate and Operations Contact
John L. Roberson (615)455-2611, x-5294
ASTF MS900

Toshiba Corporation (TIS)

TELEX: J22587

Installation Delegate
Kenjo Yoshimura 044-541-1743
TIS Division
1-1, Shibaura
Minato-Ku, Tokyo, 105
Japan

Technical and Operations Contact
Kyosuke Tsuruta 044-541-1743
TIS Division
Computer Center
72 Horikawa-cho, Saiwai-ku
Kawasaki 210, Japan

U.C. Berkeley (UCBERK)
259 Evans Hall
Berkeley, CA 94720

<u>Installation Delegate</u>

Urszula Frydman	(415)643-6097

United Information Services, Inc. (UISCO)
2525 Washington
Kansas City, MO 64108

<u>Installation Delegate and Operations Contact</u>

Nate Losapio	(816)221-9700, x.6535

<u>Technical Contact</u>

John McComb	(816)221-9700

University of Illinois at Urbana-Champaign (UIUCNCSA)
National Center for Supercomputing Applications
1011 W. Springfield
Urbana, IL 61801

<u>Installation Delegate and Technical Contact</u>

Win Bernhard	(217)333-8049

<u>Operations Contact</u>

Mike Smith	(217)244-0708

University of London (ULCC)
Computer Center
20 Guilford Street
London WC1N 1DZ
England

TELEX: 8953011

<u>Installation Delegate</u>

Richard Field	(01)4058400

<u>Technical Contact</u>

Harald Kirkman

<u>Operations Contact</u>

Lawrie Tweed

University of Minnesota Computer Center (MINN)
2520 Broadway Drive
Lauderdale, MN 55113

<u>Installation Delegate</u>

John Sell	(612)373-7878

<u>Technical Contact</u>

Linda Gray	(612)376-5603

<u>Operations Contact</u>

Elizabeth Stadther	(612)373-4920

University of Texas System (UTXCHPC)
Center for High Performance Computing
Commons Building, Balcones Research Center
PO Drawer S
Austin, TX 78713-7388

Installation Delegate
Charles Warlick (512)471-2472

Technical Contact
William Bard (512)471-2472

Operations Contact
Robert Baker (512)471-2472

University of Toronto (UTORONTO)
Computing Services
255 Huron St.
Toronto, Ontario M5S1A1
Canada

Installation Delegate and Technical Contact
Edmund West (416)978-4085
MPP 331

Operations Contact
Bob Chambers (416)978-7092
MP 350

Westinghouse Electric Corporation (WESTESCC)
Energy Systems Computer Center
P. O. Box 355
Pittsburgh, PA 15146

TELEX: 234992503 USA

Installation Delegate
Robert Price (412)374-5826
MNC 206E

Technical Contact
Jerry Kennedy (412)374-4399
Nuclear Center 180

Operations Contact
R.W. Kunko (412)374-4674
Fran Pellegrino

Zentralstelle fur das Chiffrierwesen (ZfCh)
am Nippenkreuz 19
Bonn 5300
West Germany

Installation Delegate
Engelbert Tausch

ZeroOne Systems, Inc. (ZERO)
NASA Ames Research Center
M/S 233-3
Moffett Field, CA 9[illegible]35

Installation Delegate, Technical and Operations Contact
Glenn Lewis (415)694-6550

University of London Computer Centre,
20 Guilford Street,
London, WC1N 1DZ.
England.
Tel: 01-405-8400
Telex: 8953011

CRAY USER GROUP

CALL for PAPERS
NEW YORK
April 21-25, 1987
THEME: Reliability of Software and Hardware

NAME:	
ORGANIZATION:	
ADDRESS:	
TELEPHONE:	TELEX:
ELECTRONIC MAIL ADDRESS:	
TITLE OF PAPER:	
TWO OR THREE SENTENCE ABSTRACT:	
EQUIPMENT REQUIRED OTHER THAN 35MM SLIDE PROJECTOR OR OVERHEAD PROJECTOR:	

SUGGESTED SESSION:

GENERAL SESSION	___	LANGUAGES	___
I/O	___	LIBRARIES	___
FRONT ENDS	___	MULTITASKING	___
NETWORKING	___	OPTIMIZATION	___
OPERATIONS	___	PERFORMANCE EVALUATION	___
OPERATING SYSTEMS	___	APPLICATIONS	___
GRAPHICS	___	DATA MANAGEMENT	___

RETURN BY 1 December 1986 to:
DAVID LEXTON at the above address

OTHER ____________________