NASA Technical Memorandum 101953

Initial Operating Capability for the Hypercluster Parallel-Processing Test Bed

Gary L. Cole and Richard A. Blech
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio

and

Angela Quealy
Sverdrup Technology, Inc.
NASA Lewis Research Center Group
Cleveland, Ohio

Prepared for the
Fourth Conference on Hypercubes, Concurrent Computers, and Applications cosponsored by the U.S. Department of Energy (Applied Mathematical Sciences Program), Strategic Defense Initiative Organization (Office of Innovative Science and Technology), Joint Tactical Fusion Program Office, U.S. Air Force (Electronic Systems Division), Air Force Office of Scientific Research, and NASA Ames Research Center
Monterey, California, March 6–8, 1989

NASA

INITIAL OPERATING CAPABILITY FOR THE HYPERCLUSTER PARALLEL-PROCESSING TEST BED

Gary L. Cole and Richard A. Blech
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

and

Angela Quealy
Sverdrup Technology, Inc.
NASA Lewis Research Center Group
Cleveland, Ohio 44135

ABSTRACT

The NASA Lewis Research Center is investigating the benefits of parallel processing to applications in computational fluid and structural mechanics. To aid this investigation, NASA Lewis is developing the Hypercluster, a multiarchitecture, parallel-processing test bed. This paper describes the initial operating capability (IOC) being developed for the Hypercluster. The IOC will provide a user with a programming/operating environment that is interactive, responsive, and easy to use. The IOC effort includes the development of the Hypercluster operating system (HYCLOPS). HYCLOPS runs in conjunction with a vendor-supplied disk operating system on a front-end processor (FEP) to provide interactive, run-time operations such as program loading, execution, memory editing, and data retrieval. Run-time libraries, that augment the FEP FORTRAN libraries, are being developed to support parallel and vector processing on the Hypercluster. Special utilities are being provided to enable passage of information about application programs and their mapping to the operating system. Communications between the FEP and the Hypercluster are being handled by dedicated processors, each running a message-passing kernel, (MPK). A shared-memory interface allows rapid data exchange between HYCLOPS and the communications processors. Input/output handlers are built into the HYCLOPS-MPK interface, eliminating the need for the user to supply separate I/O support programs on the FEP.

INTRODUCTION

NASA Lewis relies heavily on computational fluid mechanics (CFM) and computational structural mechanics (CSM) to simulate the behavior of aerospace propulsion systems and components. The computer codes are computationally intensive, and solution times range from hours to days, even on today's supercomputers. Computing times and memory requirements will increase rapidly as the need for more accurate and complex simulations grows.

To make CFM/CSM codes practical for applications such as propulsion system design, analysis, and on-line support of experiments, methods must be found to speed up solutions. Parallel processing technology offers potential for significant reductions in the computation time of these problems. In recent years, a number of different architectures have been proposed that generally fall into the categories of shared or distributed memory machines. At present it is not clear which types or combinations of architectures will be most suitable for the propulsion applications. Also, it may be necessary to develop new algorithms to take full advantage of promising multiprocessor architectures.

In order to assess the benefits of parallel processing to computational mechanics problems, NASA Lewis is conducting studies both in-house (Refs. 1 and 2) and through support of university research. To aid these studies, NASA Lewis researchers are developing the Hypercluster, a multiarchitecture, parallel-processing test bed. The Hypercluster is not

meant to compete with commercial parallel processors, but rather to provide a low-cost, unified approach to investigating combinations of parallel algorithms and architectures for a variety of applications. It will also provide insight into the suitability of emerging commercial parallel processors for these applications.

The Hypercluster architecture is similar to that of a hypercube, except that each node consists of multiple scalar and/or vector processors, communicating through shared memory. The result is a combination of both shared and distributed memory architectures, which allows emulation of a wide variety of architectural configurations. A commercial front-end processor (FEP) serves as the user interface to the Hypercluster.

Considerable effort is being devoted to making the Hypercluster as user oriented as possible. An initial operating capability (IOC) has been defined to provide basic programming and operating functions, as well as other capabilities. The IOC requires the development of new and modified software tools that reside on the FEP. The IOC is designed to provide a convenient, versatile programming and operating environment and to make parallel processing transparent to the user.

Past in-house experience with parallel processing hardware and software is being used as a basis for the IOC development. First-generation multiprocessor hardware (Ref. 3) and software (Refs. 4 to 8) were developed as part of the real-time multiprocessor simulator (RTMPS) project. The RTMPS was designed for real-time solution of one-dimensional, ordinary differential equation models of air-breathing propulsion systems. The IOC effort is extending those capabilities to allow solution of models, characterized by multidimensional, partial differential equations.

This paper describes the IOC design, planned capabilities, and development approach. An overview of the Hypercluster test bed and the FEP is provided first, followed by a description of the major IOC software efforts. The current status of the project and some anticipated enhancements are also presented.

HYPERCLUSTER SYSTEM CONFIGURATION

The general Hypercluster system configuration is shown in Fig. 1. The major hardware elements are the Hypercluster test bed, and a front-end processor (FEP). The FEP is the user's communication link to the Hypercluster. It has the usual peripheral equipment for storage and display (terminals, disk drives, and printers).

The Hypercluster architecture consists of clusters of processors at nodes, with the nodes interconnected by links in a hypercube fashion. A four node version is currently being implemented and is shown schematically in Fig. 2. The communication links (CLs) allow communication between nodes and consist of two control processors (CPs) communicating through dual-ported memory. An identical link is used to connect a Hypercluster node to the FEP. More than one node can be linked to the FEP if desired. Each node can consist of any number and combination of processors. Scalar and vector processors (SPs and VPs) are currently being used. The VPs act as peripherals to the SPs. Processors within a node communicate through shared memory, which may be dual-ported memory on the processor board itself, or a separate memory board. The combination of distributed and shared memory allows for emulation of a wide variety of architectures, either through software by the way it is programmed, or through hardware by rearranging the resource complement of each node. The SPs and VPs are used to perform application program computations. The CPs could also be used but this may degrade their performance as communications processors. The CP's main function is to coordinate communications over the links and supervise the operation of processors within a node. This can be done without interrupting the SPs or VPs, which may be busy with application programs. All Hypercluster components are commercially-available, except for the communication-link dual-port memories. Additional details concerning the Hypercluster hardware are given in Ref. 9.

Executive software, referred to as the message-passing kernel (MPK), runs on each CP to perform the communications and supervisory functions. The MPK, which also runs on each SP, efficiently routes information through the Hypercluster. It uses fast shared-memory communication whenever possible, or a message-passing protocol if necessary. The MPK, developed in-house, uses a layered approach to define the various kernel elements. The outermost layer consists of interfaces to the Hypercluster operating system HYCLOPS, which resides on the FEP, allowing interaction

between the FEP and the Hypercluster. A special in-house utility tests the Hypercluster hardware (memory, interrupts, etc.), loads the MPK from FEP disk files, and initializes it for the desired configuration. Loading the MPK, rather than having it reside on PROMs, provides greater flexibility for debugging and upgrading the MPK and allowing for Hypercluster configuration changes. The MPK is described in detail in Ref. 10.

The FEP is a commercially-available Motorola VME-based development system with a 68020 processor. The FEP-resident disk operating system (DOS) is a version of Motorola's VERSAdos that provides the usual utilities, such as an assembler, linkage editor, text editor, and file handling services. A FORTRAN 77 compiler and associated libraries are used to develop application programs. The DOS also provides task and memory management services and a multitasking capability, all of which provide essential support to the operating environment. A Hypercluster operating system HYCLOPS, that runs in conjunction with the DOS, is being developed to provide run-time operations such as program loading, execution control, and data handling. Data and information exchanges between the Hypercluster and FEP take place over the FEP/node communication links.

DESCRIPTION OF IOC CAPABILITIES

The goal of the IOC effort is to provide a user-oriented environment for programming and operating the Hypercluster system. This means developing software tools that are easy to learn and use, are interactive to provide flexibility, and make the parallel processing aspects of the Hypercluster transparent to the user. The software must be developed in a manner that will be compatible with a test-bed system. That is, the tools must be easy to debug and allow upgrade/expansion of their capabilities. The software needed to support these objectives is shown in Fig. 3. New software being developed for the IOC is designated by shaded items. Existing software that requires modification for the IOC is designated by items with hatching. The remaining software is resident on the FEP or is generated as part of the programming process.

Programming Environment

An application program begins with the development of source code in FORTRAN, the only language currently supported by the IOC. Source code can be created on the FEP or ported to the FEP from mainframes via local area networks. Data flow analyses, such as vectorizing and partitioning the code into parallel tasks, must be done manually, since the current FEP-resident compiler does not have those capabilities. However, compilers on NASA Lewis mainframe computers are available to aid the user in that process. The user is currently responsible for targeting programs to particular nodes and processors.

In order to support operating environment functions, data base files are required that describe the application program(s) to the operating system HYCLOPS. A data-base approach was taken because a similar technique was used successfully with the RTMPS project (Refs. 3 to 7). The data base files contain records of information that describe the programs and their variables. A typical record for a program variable would include information such as its data type and precision (e.g., real/integer, single/double), starting address in memory, number of dimensions and dimension size. Two utilities are needed to create the necessary data base files. The mapping utility sets up shared memory, if required, for programs on the same node but different processors and maps the parallel paths onto the hardware. The mapping utility also creates files to simplify and automate the Hypercluster loading process in the operating environment. A data-base utility creates files that support HYCLOPS interactive functions, such as the modification and display of program variables. Both utilities are designed to prompt the user for required information.

Existing FEP-resident utilities compile, assemble and link the source programs to produce the executable application load (object) modules. The linker automatically calls in three support libraries. The FORTRAN library is required for mathematical functions, I/O support, and run-time error handling. In order to generate object code that is executable on the Hypercluster processors it was necessary to obtain and modify an assembly source code version of the library. A run-time library of

vector processing operations and error handling was acquired with the vector boards purchased for the Hypercluster. These library routines also required modification. The parallel processing library provides procedures to support data transfers and synchronization between nodes and special I/O. This library takes advantage of services provided by the MPK. The FORTRAN, vector-processing, and parallel-processing libraries eliminate the need for any user-supplied procedures. The user simply includes the required library calls in the FORTRAN source code.

Operating Environment

Once programming is complete, operating functions are required to load and execute the object module(s) and to retrieve application results. There are no FEP-resident utilities to support these functions. To provide these functions, a new Hypercluster operating system HYCLOPS is being developed. It runs on the FEP in conjunction with the resident DOS. The HYCLOPS multitask design, providing the necessary functions to achieve the IOC goals, is shown in Fig. 4. There are three major tasks. Shared memory provides for communications required between tasks and the FEP, as well as for storage of application results and advisory messages.

The interactive task provides the user with the functions necessary for executing the application programs. Its menus and prompts make it easy to learn and use, and virtually eliminates the need to know FEP-DOS commands. Responses to prompts can be entered interactively via the keyboard or "automatically" via predefined files. For example, the user can designate file names and the node and processor destinations to load executable object modules. Or the user can select the load function, which will automatically load the modules from database files. To do this, only the application program name is required. A self-documenting session history records all user entries, as well as pertinent task prompts, and saves messages from the message advisory task. The session history was a powerful feature included in RTMPOS (Refs. 6 and 7). The file is useful for reviewing session progress and coordinating it with results. It can also be used as an input file to HYCLOPS to recreate the session without making manual keyboard entries. The interactive task has a number of features to minimize response time. The application data base is read into memory for faster access than from disk files. The MPK message-passing protocol is used to exchange data between the FEP and the Hypercluster. Messages between the FEP and specific Hypercluster processors are directed through the nearest FEP link if more than one exists. Shared memory between the FEP and CPs on the FEP bus results in direct transfer of data between the FEP and the dual-port interface memory. It also results in "automatic" conversion between bytes of information and the desired data types, such as real numbers, which is discussed in the next section.

The interactive task supports the following user functions. As described above, application object modules can be loaded interactively or automatically via the application data base. If the auto mode is selected, the application data base is first loaded into FEP memory. A data base manager provides functions for editing and manipulating the data base. Values for initializing selected program variables will be included in the data base and set at run time. Modification and display of memory locations anywhere in the Hypercluster can be accomplished by means of the memory editor function. When an application data base is used, program variables can be specified to the memory editor symbolically by name. This facilitates debugging of programs. Once loaded and initialized, the application programs can be executed on the Hypercluster interactively. The execution mode manager can be invoked from any menu, allowing the user to RUN, STOP, or RESUME execution. Selection of RUN causes all loaded processors throughout the Hypercluster to begin execution at the program entry point. STOP causes all loaded processors to stop execution. The RESUME mode allows all loaded processors to resume execution from the point at which they halted due to a STOP. The mode manager also allows the user to display the current RUN/STOP status of all Hypercluster processors. Another major interactive task function is assignment of files to retrieve application results. A maximum of 10 FORTRAN output units can be assigned by the user at run time. The user specifies the FORTRAN unit number, the file name to be written to, and the file type (i.e., formatted or unformatted). If an existing file is specified, the user has the option of overwriting it, appending to it, or specifying a new file name. A

user-transparent data advisory task, described below, is started by HYCLOPS to support each output unit. The tasks are terminated by a FORTRAN CLOSE command included in the application program or interactively by the user. Display of active units and associated files is menu selectable. The interactive task can be terminated and restarted without affecting the data advisory tasks, which will not be interrupted.

A separate message advisory task retrieves error messages originating in the Hypercluster. The messages are displayed on a user-selected message device and saved in the session history file. The task services system errors, such as a bus error, as well as run-time errors supported by the vector-processing, parallel-processing, and FORTRAN libraries. An example of the latter would be a divide by zero. Depending on the severity of the error, the MPK can halt all Hypercluster processors. In that case, a register dump is produced for the processor having the error. This task functions automatically and is transparent to the user.

In order to support retrieval of applicaton results, a generic data advisory task is created each time the user assigns a FORTRAN unit to a file. If programs on different processors have duplicate unit numbers, the user is required to coordinate the write statements to avoid unwanted interlacing of data, (if necessary). This can be done by making appropriate calls to the parallel-processing library. The MPK transfers results from the originating processor to a data buffer in the application results data segment of shared memory. The MPK places a pointer to the buffer in the data advisory task's queue. The task transfers the data to the disk file using the FEP-DOS I/O services. Once the transfer is complete, the task clears the queue and makes the data buffer available for reuse. This approach for retrieving results eliminates the need for the user to supply special output programs on the FEP.

IOC SOFTWARE DEVELOPMENT APPROACH

The new software tools shown in Fig. 3 are being developed in three phases - design, programming, and testing. Because the database and mapping utilities and HYCLOPS are coupled through the data base files, these software efforts must be closely coordinated. To minimize development time, the utilities and HYCLOPS are designed to take advantage of as much FEP-resident software as possible. The programming environment design uses command/control files to automate the code generation process, where possible, and will allow advanced compilers and data-flow-analysis tools to be incorporated, as they become available. As shown in Fig. 3, the HYCLOPS design makes use of task initialization files. These are text files that can be easily edited to account for changes in operating environment features without having to reprogram/recompile HYCLOPS tasks. For example, the message advisory task uses a file that contains the message-advisory shared-memory-segment attributes, including starting address, size, and number of message buffers.

A top-down programming approach is being used so that IOC software can be expanded and easily modified. Pascal is used as the programming language, as much as possible, to maximize portability of the IOC to other FEPs. The FORTRAN and vector-processing libraries were supplied by the vendor in assembly language. The parallel-processing library is programmed in assembly language to maximize processing speed on the Hypercluster. Some HYCLOPS procedures are programmed in assembly language because standard Pascal does not support certain operations, such as writing to specific memory addresses. All assembly language routines are specific to Motorola 68000-series processors. But most are relatively simple and can easily be retargeted to hardware from other manufacturers. Typical software interfaces are shown for HYCLOPS in Fig. 5. Actual proportions of Pascal and assembly code is not represented. HYCLOPS is primarily composed of a Pascal kernel. An assembly language interface is required for HYCLOPS to initiate a message to the MPK. This is done by writing to specific memory addresses in the CP that links the FEP to the Hypercluster. Sometimes the message will be a request for data from the Hypercluster (e.g., the value of a program variable). In that case, HYCLOPS provides a return address in FEP memory that corresponds to a Pascal record of the required data type and precision (e.g., real, single). This eliminates the need for a conversion between the bytes of information being returned and the required data types. The same approach is used when sending data to the Hypercluster. Both Pascal and assembly language interfaces are required between HYCLOPS and the FEP-resident DOS. Pascal is used mainly to interface with the DOS I/O utilities. Assembly language is used to interface with DOS utilities such as task and memory management.

Testing and debugging of IOC software is done to the extent possible as programming proceeds. Simple FORTRAN programs are being written to test the three libraries supporting the compiler. Each HYCLOPS function is tested as it is developed and added to the interactive task. A representative CFM application will be selected to test and demonstrate the entire IOC, before making the Hypercluster system generally available to users. The choice of a relatively simple code is important to prevent massive calculations or other program complications from interfering with testing of data transfers, vector operations, etc. Demonstration of the benefits of parallel/vector processing is not a primary objective of this test.

CONCLUDING REMARKS

Design and development of an initial operating capability (IOC), that provides user-oriented programming and operation of the Hypercluster parallel-processing test bed, has been described. The Hypercluster architecture, coupled with the IOC, should provide researchers in computational mechanics with a unique facility for exploring the benefits of advanced algorithms and computer architectures to their applications. The IOC effort requires development of new and modified software tools that reside on a front- end processor (FEP) and make use of the resident disk operating system (DOS) facilities.

Sufficient software tools are currently in place to begin programming applications in FORTRAN. Libraries of procedures to support FORTRAN functions, vector-processing, and parallel-processing have been developed/ modified, thus eliminating the need for user-supplied procedures. The user simply includes the required library calls in the FORTRAN source code. The new Hypercluster operating system, HYCLOPS, currently has capabilities for interactively loading and executing application programs. Data advisory tasks can be assigned to FORTRAN output units at run time to retrieve application results, eliminating the need for any user-supplied output support programs. A HYCLOPS message advisory task retrieves and displays system and run-time error messages from the Hypercluster to the user.

Additional capabilities are still being added to the programming environment. The new parallel-processing library is in the process of being tested. Data-base and mapping utilities are being added to simplify loading of applications and to support interactive capabilities being added to HYCLOPS (e.g., symbolic editing of program variables at run time). Command/control files for "automating" the programming process are being developed. HYCLOPS capabilities being added include a unique self-documenting session history file and optional input of user entries from the keyboard or predefined disk files. The session file can also be used as input to HYCLOPS to recreate the session without making manual keyboard entries. The additional capabilities are planned for completion by the second quarter of 1989.

Testing of computational fluids applications has already begun. Since the Hypercluster is a test-bed environment, it is expected that refinements will be made based on user experience, as well as enhancements and additions based on the availability of new/advanced software tools (e.g., compilers with vectorizing and partitioning capabilities). The need for on-line graphical-display of application results must be addressed. The possibility of conversion to a more standard FEP operating system, such as UNIX, is being investigated. This would provide portability of the programming/ operating environment to a variety of workstations, which in turn would increase the availability of graphics and better data-flow-analysis tools. Although multiple FEPs can be connected to the Hypercluster test bed, it is currently viewed as a single-user system. Neither HYCLOPS nor the MPK provide resource management, but could be modified to do so. Enhancement of run-time debug capability, such as the ability to more easily set and remove break points at strategic instructions, should also be addressed.

REFERENCES

1. E.J. Milner, R.A. Blech, and R.V. Chima, Time-Partitioning Simulation Models for Calculation on Parallel Computers; NASA TM-89850, 1987.
2. R.A. Mulac, M.L. Celestina, J.J. Adamczyk, K.P. Misegades, and J.M. Dawson, The Utilization of Parallel Processing in Solving the Inviscid Form of the Average-Passage Equation System for Multistage Turbomachinery, AIAA Paper 87-1108, June 1987.
3. R.A. Blech and A.D. Williams, Hardware Configuration for a Real-Time Multiprocessor Simulator, NASA TM-88802, 1986.

4. D.J. Arpasi, RTMPL - A Structured Programming and Documentation Utility for Real-Time Multiprocessor Simulations, NASA TM-83606, 1985.
5. D.J. Arpasi, Real-Time Multiprocessor Programming Language, (RTMPL) - Users Manual, NASA TP-2422, 1985.
6. G.L. Cole, Operating System for a Real-Time Multiprocessor Propulsion System Simulator, NASA TM-83605, 1984.
7. G.L. Cole, Operating System for a Real-Time Multiprocessor Propulsion System Simulator - Users Manual, NASA TP-2426, 1985.
8. D.J. Arpasi, and E.J. Milner, Partitioning and Packing Mathematical Simulation Models for Calculation on Parallel Computers, NASA TM-87170, 1986.
9. R.A. Blech, The Hypercluster: A Parallel Processing Test-Bed Architecture for Computational Mechanics Applications, NASA TM-89823, 1987.
10. R.A. Blech, A. Quealy, and G.L. Cole, A Message-Passing Kernel for the Hypercluster Parallel-Processing Test Bed, NASA TM-101952, 1989. Proceedings of the Fourth Conference on Hypercubes, Concurrent Computers, and Applications.(to be published Monterey, CA, Mar. 6-8, SIAM, Philadelphia, PA, 1989).

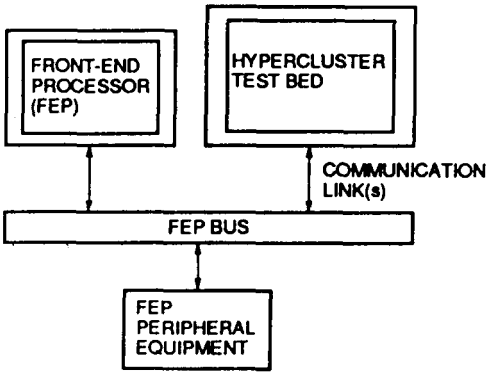

Figure 1. - Hypercluster system configuration.

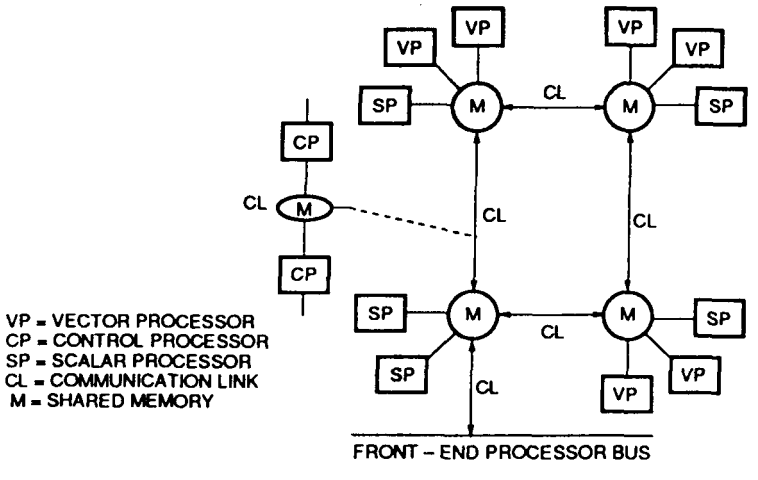

Figure 2. - Hypercluster test bed architecture.

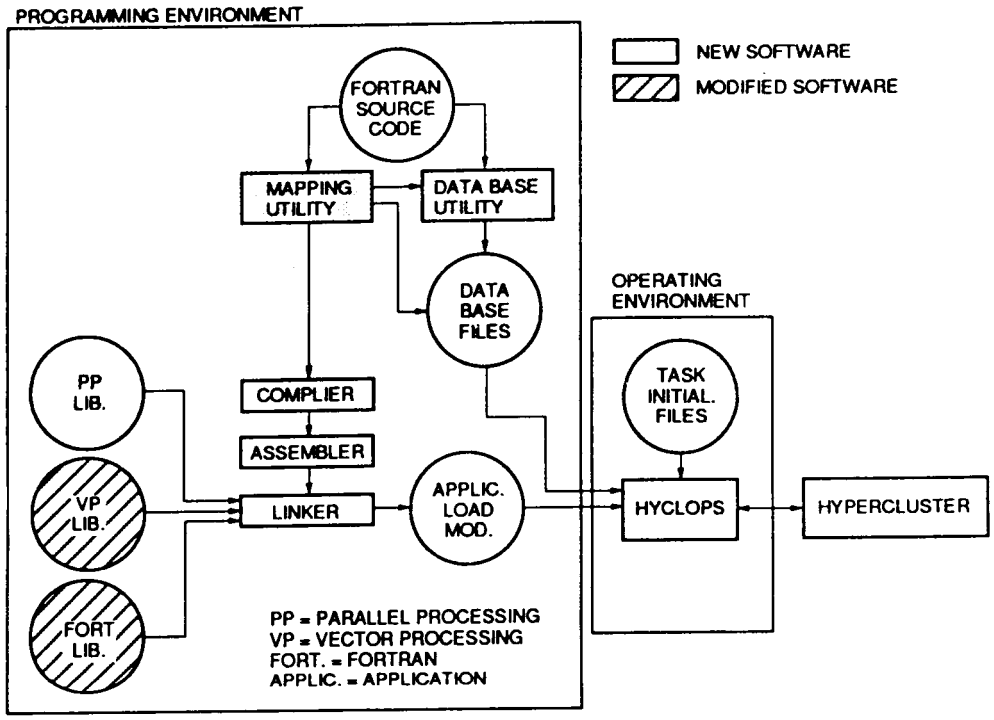

Figure 3. - IOC programming/operating environment.

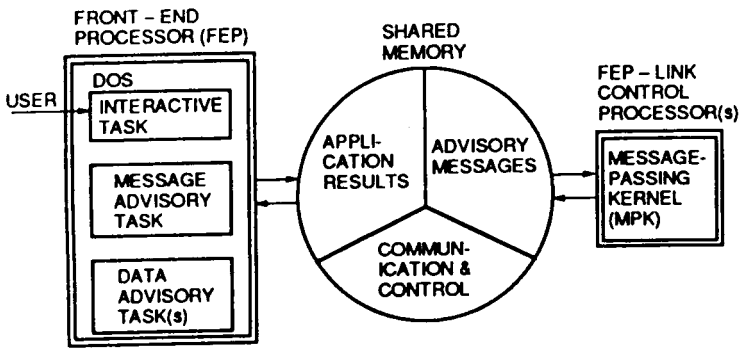

Figure 4. - Operating environment (HYCLOPS) multitask structure.

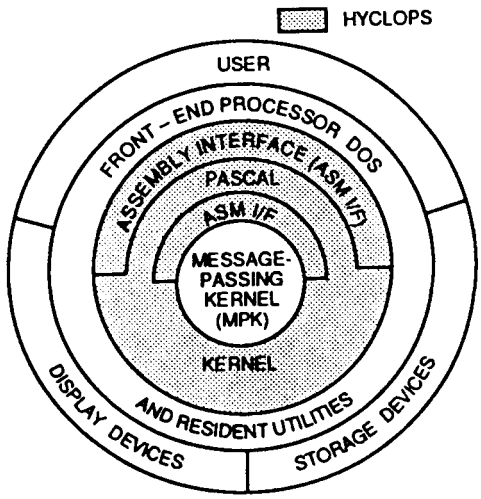

Figure 5. - Hypercluster operating system (HYCLOPS) software interfaces.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-101953		

4. Title and Subtitle	5. Report Date
Initial Operating Capability for the Hypercluster Parallel-Processing Test Bed	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Gary L. Cole, Richard A. Blech, and Angela Quealy	E-4647
	10. Work Unit No. 505-62-21

9. Performing Organization Name and Address	11. Contract or Grant No.
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191	
	13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, D.C. 20546-0001	

15. Supplementary Notes

Prepared for the Fourth Conference on Hypercubes, Concurrent Computers, and Applications, cosponsored by the U.S. Department of Energy (Applied Mathematical Sciences Program), Strategic Defense Initiative Organization (Office of Innovative Science and Technology), Joint Tactical Fusion Program Office, U.S. Air Force (Electronic Systems Division), Air Force Office of Scientific Research, and NASA Ames Research Center, Monterey, California, March 6-8, 1989. Gary L. Cole and Richard A. Blech, NASA Lewis Research Center; Angela Quealy, Sverdrup Technology, Inc., NASA Lewis Research Center Group, Cleveland, Ohio 44135.

16. Abstract

The NASA Lewis Research Center is investigating the benefits of parallel processing to applications in computational fluid and structural mechanics. To aid this investigation, NASA Lewis is developing the Hypercluster, a multi-architecture, parallel-processing test bed. This paper describes the initial operating capability (IOC) being developed for the Hypercluster. The IOC will provide a user with a programming/operating environment that is interactive, responsive, and easy to use. The IOC effort includes the development of the Hypercluster Operating System (HYCLOPS). HYCLOPS runs in conjunction with a vendor-supplied disk operating system on a Front-End Processor (FEP) to provide interactive, run-time operations such as program loading, execution, memory editing, and data retrieval. Run-time libraries, that augment the FEP Fortran libraries, are being developed to support parallel and vector processing on the Hypercluster. Special utilities are being provided to enable passage of information about application programs and their mapping to the operating system. Communications between the FEP and the Hypercluster are being handled by dedicated processors, each running a Message-Passing Kernel, (MPK). A shared-memory interface allows rapid data exchange between HYCLOPS and the communications processors. Input/output handlers are built into the HYCLOPS-MPK interface, eliminating the need for the user to supply separate I/O support programs on the FEP.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Operating Systems; Multiprocessor programming; Multiprocessors; Hypercube; Parallel processors	Unclassified - Unlimited Subject Category 62

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No of pages	22. Price*
Unclassified	Unclassified	10	A02

*For sale by the National Technical Information Service, Springfield, Virginia 22161