Title
USING COOPERATIVE SIMULTANEOUS PARALLELISM IN NONHOMOGENEOUS MICROCOMPUTER CLUSTERS

Author
Meng, J.

Publication Date
1984-10-01

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

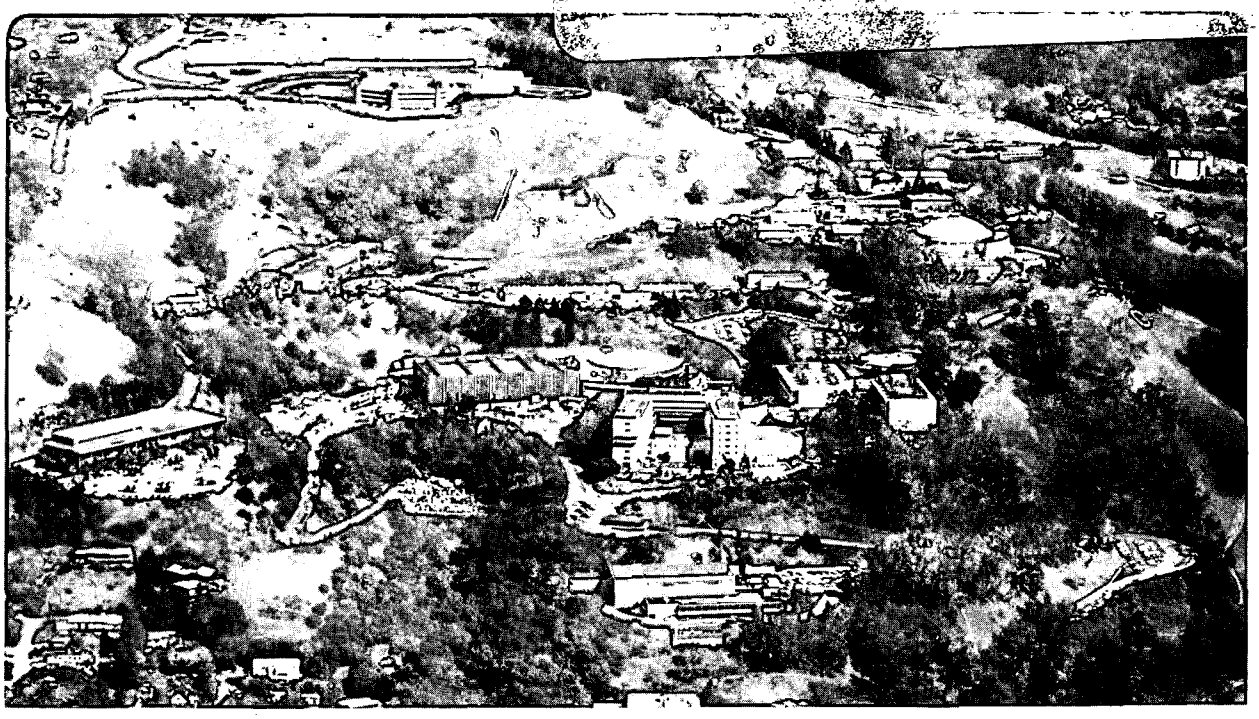
Engineering Division

Presented at the International Society for
Mini and Microcomputers International Symposium
"Applications of Microcomputers", New York, NY,
October 22-24, 1984

USING COOPERATIVE SIMULTANEOUS PARALLELISM
IN NONHOMOGENEOUS MICROCOMPUTER CLUSTERS

J. Meng

October 1984



Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

LBL-17801

DISCLAIMER

USING COOPERATIVE SIMULTANEOUS PARALLELISM IN NONHOMOGENEOUS MICROCOMPUTER CLUSTERS

John Meng

Lawrence Berkeley Laboratory
University of California
Berkeley, California U.S.A. 94720

ASTRACT

Based on our experience controlling different types of microprocessors in clusters through the central processor-memory connection, and based on our successful parallel operation of a cluster of minicomputer central processors, a MIDAS-like arrangement of different microcomputers in a cluster is useful in dealing with a variety of problems. The MIDAS-like arrangement refers to master-slave control of the cluster by a minicomputer and to data being passed through the cluster by connecting prefilled memories into and out of processor address space. We discuss how to connect the hardware into a cluster and conclude with descriptions of how to apply the hardware to selected diverse applications.

INTRODUCTION

The original MIDAS cluster[1], a demonstration device, is achieving a linear speedup in processing as a function of the number of processors in the multiprocessing structure. Figure 1 illustrates the basics of this multiprocessor. A crossbar connection controls the switching of any of sixteen memories into and out of the address space of any of up to thirteen processors. (The memory zeroing logic is treated as a processor, although not drawn as such.) Thus at any time each processor works on data in a switchable memory independent of any other processor working on data in another switchable memory. A crossbar control processor prevents conflicts and orchestrates memory and processor migrations. An independent minicomputer handles global tasks such as initialization, setup, downloading, fault and work monitoring and a limited amount of message passing from processor to processor. If a job can be duplicated and run in parallel, such as Monte Carlo calculations or code deciphering, or if finite independent data blocks must be processed, such as data analyses from large detector systems; this cluster applies 8 - 12 times the computing power of a single processor. Tree-structure problems, wherein subsequent calculations demand results from prior calculations, do not realize this much speedup, although our experience indicates substantial gains are practical. These issues are all discussed in much greater detail in the literature.[2,3,4,5]

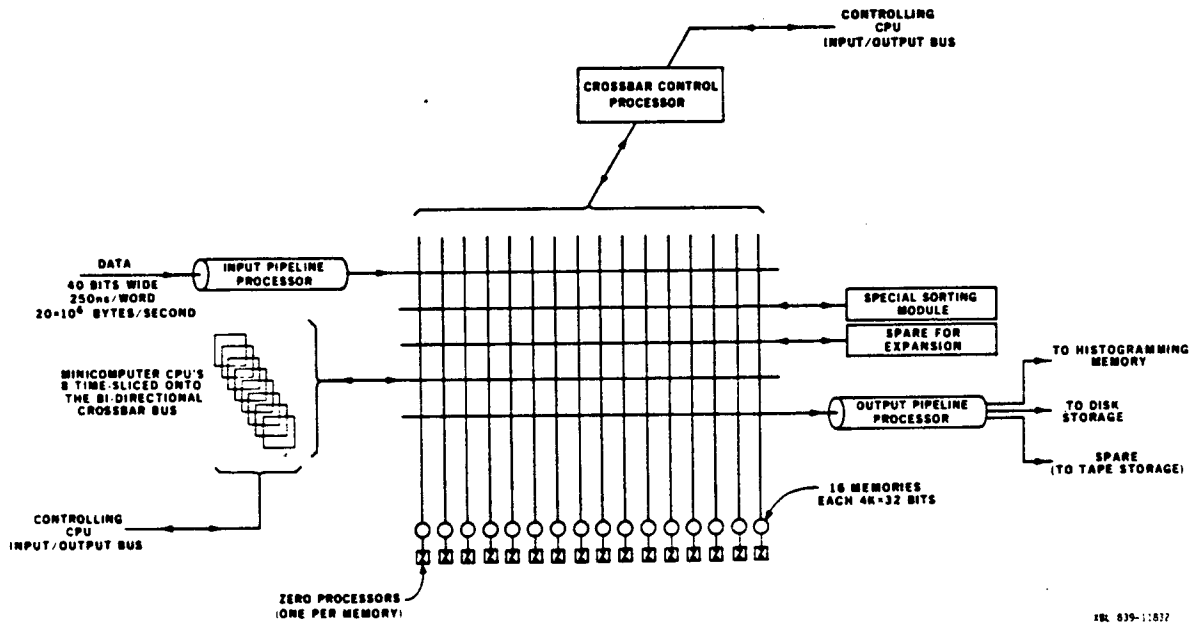


Fig. 1 In the MIDAS architecture, special purpose processors work with a cluster of minicomputer general purpose processors. Data passes through the architecture in memories connected to a preselected sequence of processors. A controlling minicomputer (not shown) is the master of this part of the system. Connections occur through what is effectively a 13 x 16 crossbar. (Zeroing logic on each memory is considered one processor.)

Our experience with the initial cluster is leading us into architectural enhancements, such as the ability to dynamically reconfigure a cluster[6], the inclusion of special purpose processors within a cluster to handle time-critical interative functions, the extension of a single cluster to its maximum practical number of processors, and the extension into multi-cluster systems for handling very large and/or very diverse functions.

From a hardware standpoint, we visualize systems with large numbers of small and efficient, but fast and powerful microcomputers and components, and conclude that parallel processing can solve problems which today, either because of the cost of processor time or because of insufficient processing speed, are not being attempted.

by firmware, hardware or software); and the pipeline processor. Our generic connection (Fig. 3) will connect to each.[7]

General purpose central processors are most useful running transported solutions--those already running elsewhere, usually programmed in some high level language, and needing a multiprocessor speedup--or in performing not-heavily-iterative functions. Monte Carlo calculations and many data analyses fit this class.

Special purpose dedicated processors are based on high speed devices with limited address space and good arithmetic abilities. Fast Fourier Transforms and display rotations are examples of applying these devices.

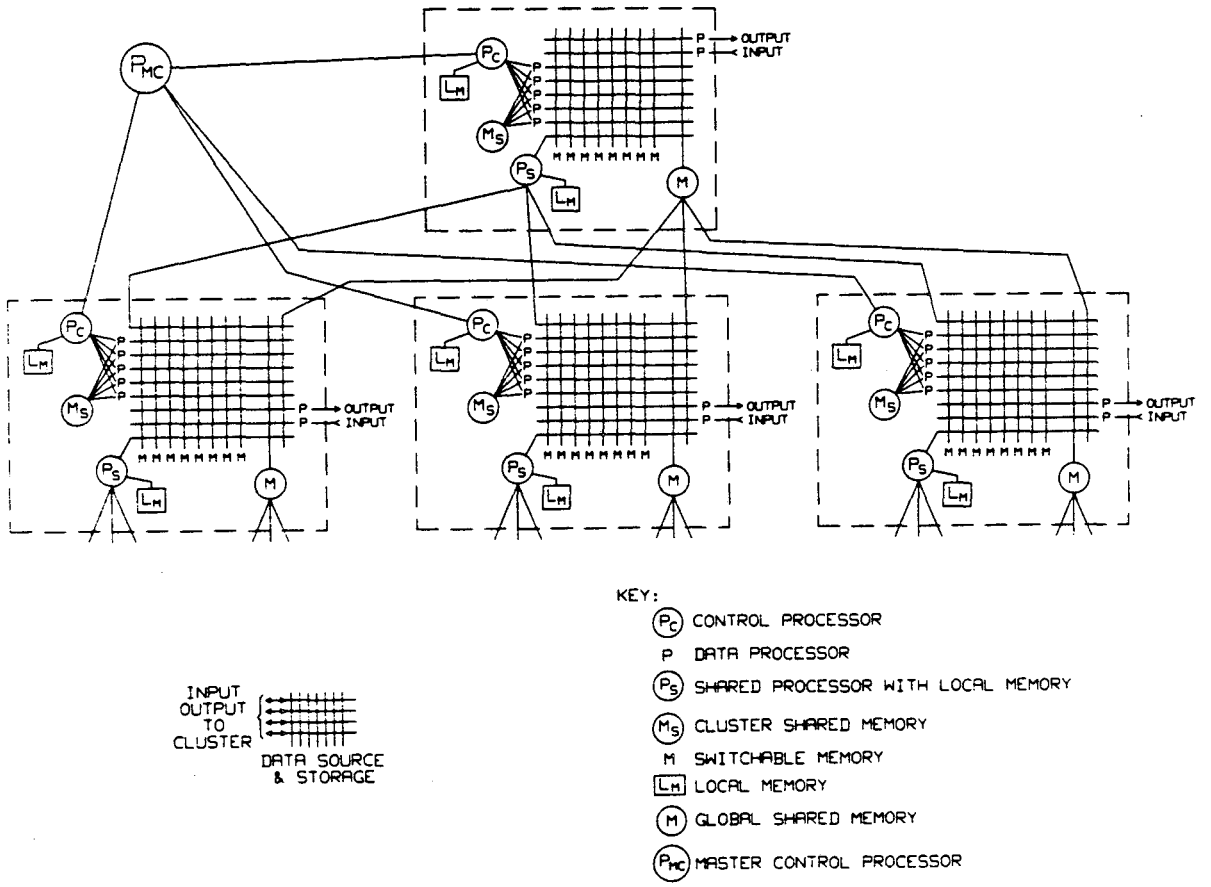


Fig. 2 This is one possible way to extend the MIDAS architecture without letting crossbar dimensions become unwieldly. Cluster levels share both processors and memories. Input and output pass through a dedicated crossbar.

CLUSTER PROCESSORS

In the extended system, cluster specialization is natural. One cluster may generate displays and communicate with the user through a variety of special devices. Another may control mechanical devices in servo loops. Still another might be doing specialized data reduction. Such diversity within a system supports specialized processor types as well as general purpose processors.

Available hardware favors three general classes of processor: the general purpose central processor; the special processor dedicated to a single function (either

Iterative bottlenecks can be brought under control using these processors. Routines occurring in critical paths of larger operations may reside in a special dedicated processor and be called upon when needed by a general purpose processor in a cluster. If the critical routine is heavily computational or depends on an iterative convergence, the special processor applies its expertise very effectively.

Pipelines are ideal for transforming or correcting long data streams. Data from a Charged-Coupled Device (CCD) detector, for example, may need systematic correction dependent on the cell being accessed. A pipeline

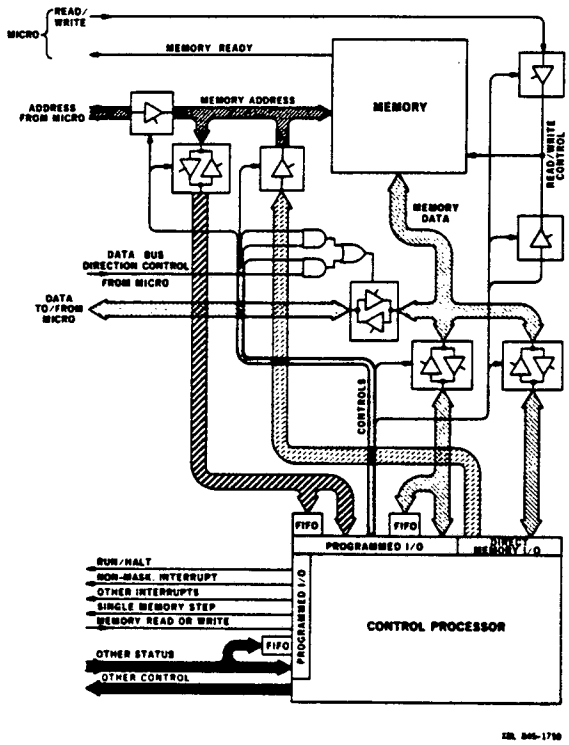


Fig. 3 This is the essence common to all microprocessor connections to the control processor. FIFO connections to high speed buses provide snapshots of bus activity to the control processor.

processor can correct for gain and offset as the data passes to storage. Recycling stored data through a pipeline can perform correlations, templating, smoothing or a host of interactive operations.

Data "threads"[6] through groups of processor types within a cluster, enabling the parsing of solutions in ways designed to maximize the speedup obtainable within the cluster.

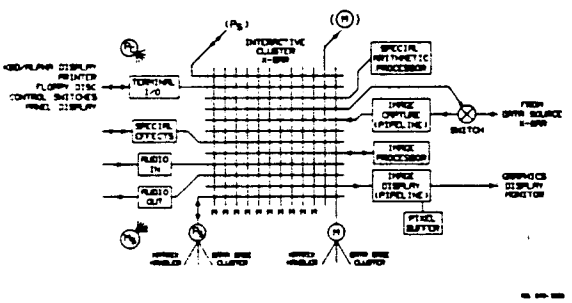


Fig. 4 Interactive clusters, as in this example, contain user dedicated special effect processors to do things like rotate images or handle voice-in-voice-out capabilities. Larger systems require the extensions.

SOLUTIONS TO SOME GENERAL PROBLEMS

We are applying the MIDAS cluster to solve a variety of specific problems.[2,5,6,8] Based on this experience and its extrapolation to newer cluster architectural enhancements and multiple clusters, the following applications seem particularly viable. Dedicated processors with special attributes such as speech and speech recognition and elaborate graphic capabilities can make systems more interactive. Furthermore, the use of dedicated processors executing simultaneously can speed responses to other system functions such as data base searches and computations. Meaningful responsive interaction with a user requires an orientation towards the available sensory inputs of the user and towards his capabilities as a signaler.[9] In Fig. 4, one processor is used to do things normally done at a terminal or with a personal computer. Audio processors are shown, as is one dedicated to unspecified special effects (lights, numeric displays, special switches, etc.). This cluster incorporates a complete imaging system. Capturing an image and writing it into a memory, m, in a coherent format are the jobs of the image capture pipeline. Rotations, shading, 3-D displays and hidden line processing are the job of the image processor, possibly in conjunction with the special arithmetic processor. The image display pipeline converts images in internal format into a pixel buffer and thence into drive signals for a graphics display. Captured images reside in switchable memories m. User signals load an available memory m and send it to the image display pipeline to select a new display or to change parameters. Similarly, an m is sent to the image processor to change parameters. The image processor releases the parameter-changing m to be zeroed and reused. It requests display-holding m's until finding the one requiring a change, makes the change and releases the m to the image display pipeline for storage as pixels and display. Extension links connect the cluster into the extended system. This cluster is a general interactive control station, potentially connecting users into a variety of systems. Its independence allows other system functions to proceed simultaneously.

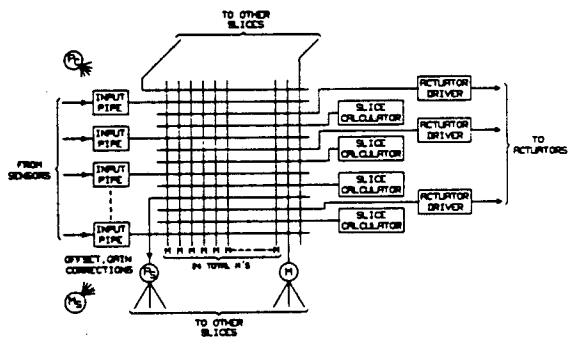


Fig. 5 The figure control processor is a functional link in a complex servo loop, and in three-dimensional systems may involve multiple clusters, each of which controls one slice. Pipeline processors normalize sensor inputs and memory movement sequences collect appropriate control data for each actuator.

The mechanical figure control system (Fig. 5) provides a means of setting and maintaining the predetermined shape of a mechanical system under changing environmental and/or controller conditions. Multiple-mirror solar energy collectors are a potential beneficiary as are multiple-mirror and segmented mirror telescopes. Sensors supply position data to input pipes for removal of statistical noise, offsets and gain variations. Memories, m, are software associated with a particular slice calculator, and are sent only to the input pipes having data required by that particular slice calculator. After passing from pipe to pipe and collecting an input from

each, m connects to the slice calculator which translates the collection of position signals into the collection of actuator signals required to correct any inaccuracies. Memory m is then passed to the actuator drivers for response. One cluster may be dedicated to a two-dimensional slice in an elaborate and/or large system, other slices having their own control cluster. The master control processor has responsibility for overseeing the relationships between individual slices and the three dimensional picture. Three dimensional corrections may pass through the cluster control processors as new offset corrections for use by input pipes. Boundary sensors for this slice may tap adjacent slice boundary sensors using cluster extenders. Without position sensors, the cluster may be controlling a flexible surface in a mechanical modeling scheme. An interactive cluster included in the system can monitor and control performance.

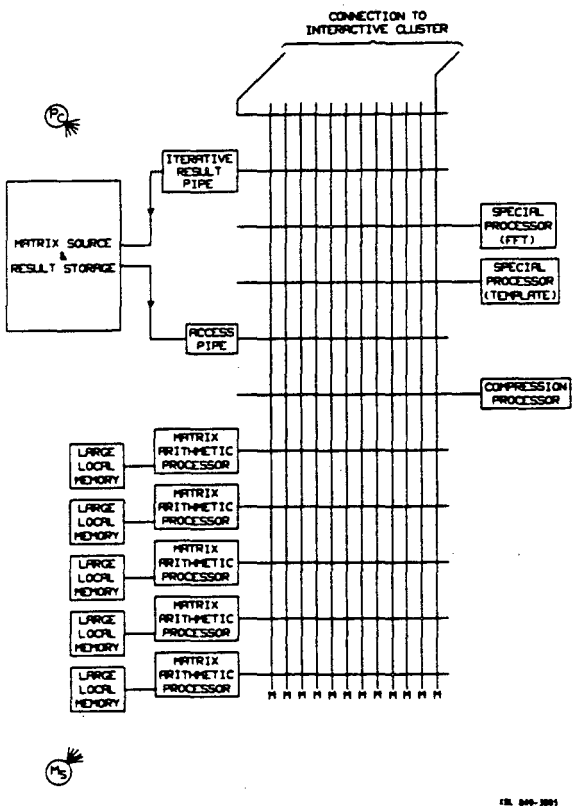


Fig. 6 Matrix problems require access of one large memory to the whole or to selected regular parts of a second large memory. The selected parts are pre-stored in local memories at each arithmetic processor, and blocks of a second matrix pass through in data transport memories, m. This configuration has potential in tomographic reconstruction and in image analysis.

We can separate many matrix-based problems into components for parallel execution. Special purpose processors, in most cases pipeline processors, do the splitting and recollection. In tomographic image reconstruction, for example,[8] submatrices may be used in the calculations, and the results combined. The same may be true with matrices derived from CCD imaging systems. The cluster in Fig. 6 stores a source matrix in a large bulk memory. A sparse matrix may be in compressed form. Interaction between matrices, as is required in tomographic reconstruction, requires the second matrix be multiply copied into large local memories on the matrix arithmetic processors. The access pipe reconstitutes and selects data from bulk storage, storing it in memories m. Full m's connect to matrix arithmetic processors where required operations are executed in parallel. Two special processors inhabit the cluster. Arithmetic processors may pass m's to these if special high-speed functions are a required part of the processing. For example, if CCD images are being analzed, a Fast Fourier Transorm (FFT) processor may be needed, or correlative template operations might be useful. These two are examples, but not the only ones possible. Finally, m's connect to a compression processor whose job is to reassemble components and pass results to the pipe for storage in bulk memory. Iterative matrix operations, such as those needed in tomographic reconstruction, may require several loops through this process.

Tree structure approaches to solving problems make use of a dedicated processor to compile statistics on frequency of use of pathways through the three, optimizing future parallel look-ahead efforts. Tree structure approaches are characterized by the necessary completion of one level of computation before the next can be started (Fig. 7). Common examples are games, such as Othello and Chess, and knowledge-based systems.

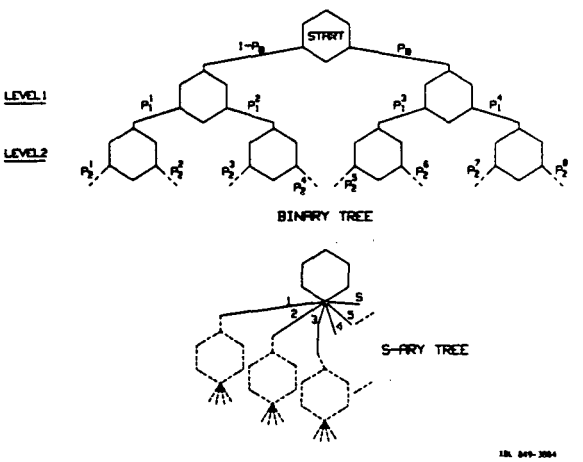


Fig. 7 In tree structures, moving down to the next level requires knowing the path from the previous level. Consequently, applying parallelism to the tree structure depends on making educated guesses of the exit path based on probabilities P_n, then calculating as far ahead in depth and in breadth as parallel resources permit. Speed-up is a function of the success of our guessing strategy and ultimately of the effectiveness of applying appropriate feedback to values of P_n.

To successfully apply parallelism to the tree structure requires looking ahead into the next level or beyond. Processor counts can go very high very quickly if one processor is dedicated to each look-ahead. This is especially true if there are more than two choices at each node. Success is critically dependent on successfully predicting which paths through the structure are most likely and applying parallel look-ahead to those paths first. In the absence of objective assignments of probabilities to the various paths, we dedicate one processor to maintaining a history of previous solutions and using this to bias future assignment of resources to look-ahead directions and depths. Figure 8 illustrates a cluster structure for handling this. Switchable memories, m, pass control information from the odds calculator and historian to each Look Ahead Processor (LAP) and pass results back to the historian. Before a run is begun, the problem is downloaded into each LAP for

parallel execution. When a LAP connects to a memory, it reads assumptions of previous results and proceeds to calculate new results, passing them, via m, back to the historian. This problem is directly extensible simply by adding more clusters of processors to widen or deepen the lookahead. The master control computer assumes overall direction in the multiple-cluster case, parsing stalks of the tree to cluster control processors and finally determining the arrival of the end of the problem.

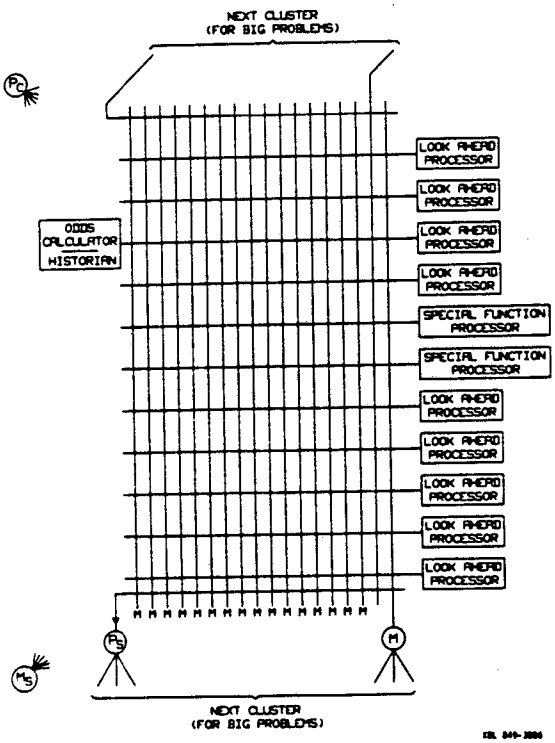


Fig. 8 Problem solutions paralleling the tree structure of Fig. 7 yield to this cluster structure. The number of look-ahead processors, in conjunction with the strategy cleverness of the odds calculator, determines the efficiency of parallel processor application to the solution.

One final class of problem is that requiring nothing more than a large number of processors under control of one central authority. Classic among these are Monte Carlo calculations and code deciphering. We feed a continuous stream of unique numbers, random in one case and trial key values in the other, into the system. It is simplest to generate these in the master control processor (Fig. 4) and pass them to cluster control processors for passage on demand to cluster processors. Other distribution schemes have been used.[10]

SUMMARY AND CONCLUSIONS

We are running some of these solutions on the MIDAS cluster, simulating configurations shown here with software and with novel exploitations of the MIDAS hardware. Specifically, tree-structured problems, Monte Carlo models, and matrix based tomographic reconstruction[4] are running, all with substantial multiprocessing speed-ups.

The future of this architecture seems to be with multiple clusters of arrays of different types of microprocessors working in concert. Scenarios have been developed and in several cases are in use applying this structure to the solution of a diversity of problems.

The MIDAS cluster architecture is expandable into a multiple-cluster architecture and based on its proven parallel processing capability is expected to multiply microprocessor computing power by a factor close to the number of microprocessors in the system. The multiple cluster architecture is generally useful in the solution of diverse and difficult problems.

ACKNOWLEDGMENTS

This work was supported by the Director's Office of Energy Research, Office of High Energy and Nuclear Physics, Division of Nuclear Physics, and by Nuclear Sciences of the Basic Energy Program of the U. S. Department of Energy under Contract Number DE-AC03-76SF00098.

A debt of gratitude to other members of the MIDAS project group must be acknowledged, especially to Creve Maples, Dan Weaver, Doug Logan, Bill Rathbun, Fong Gin and Elinor Potter.

REFERENCES

1. J Meng, D Weaver, C Maples, W Rathbun and D Logan, "An Interactive Parallel Processor for Data Analysis", IEEE Trans. on Nucl. Sci., NS-31, No. 1, 162 (1984).

2. C Maples, D Weaver D Logan and W Rathbun, "Performance of a Modular Interactive Data Analysis System (MIDAS)", Proc. of the 1983 Intl. Conf. on Parallel Processing, 514 (1983).

3. J Meng, "Multiplication of Processing Capacity with a Parallel Processor Array", Conf. Record of the Seventeenth Asilomar Conf. on Circuits, Systems and Computers (1983).

4. D Logan, C Maples, D Weaver and W Rathbun, "Adapting Scientific Programs to the MIDAS Multiprocessor System", Proc. of the 1984 Intl. Conf. on Parallel Processing.

5. C Maples, D Weaver, W Rathbun and D Logan, "The Operation and Utilization of the MIDAS Multiprocessor Architecture", Proc. of the 1984 Intl. Conf. on Parallel Processing.

6. C Maples, D Weaver, J Meng, W Rathbun and D. Logan, "Utilizing a Multiprocessor Architecture—The Performance of MIDAS", IEEE Trans. on Nucl. Sci., NS-30, 3827 (1983).

7. J Meng and F Gin, "A Control Scheme for Microcomputers Being Used in Multiprocessor Arrays", Conf. Record of the 25th Intl. Symposium on Mini and Microcomputers and Their Applications, San Francisco, CA (1984).

8. J Llacer and J Meng, "Matrix-Based Reconstruction Methods for Tomography", Proc. Nuclear Science Symposium, October 1984.

9. J Meng, "Tactile and Kinesthetic Controls for Use in Interactive Mini and Microcomputer Environments", Minicomputer Applications, Vol. 1, No. 2, 48 (1982).

11. D Logan, W. Rathbun and C Maples, "Parallel Independent Random Number Generation for Parallel Processing", In Process (1983).

This report was done with support from the Department of Energy. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the Department of Energy.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720