Title

DEVELOPMENT OF COOPERATIVE SIMULTANEOUS PARALLELISM IN A NONHOMOGENEOUS MICROCOMPUTER CLUSTER

Author

Meng, J.

Publication Date

1984-11-01

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

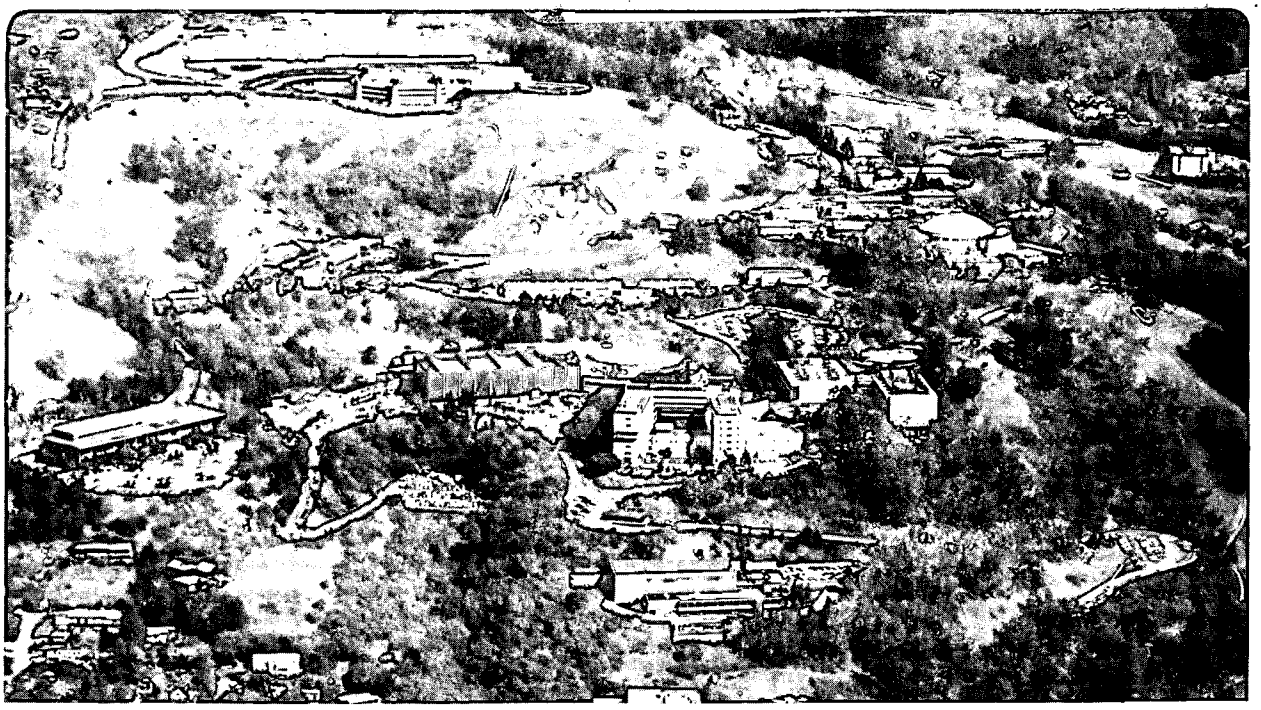
Engineering Division

Presented at the 18th Annual Asilomar Conference
on Circuits, Systems, and Computers,
Pacific Grove, CA, November 5-7, 1984

DEVELOPMENT OF COOPERATIVE SIMULTANEOUS PARALLELISM IN A NONHOMOGENEOUS MICROCOMPUTER CLUSTER

J. Meng

November 1984



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

DISCLAIMER

DEVELOPMENT OF COOPERATIVE SIMULTANEOUS PARALLELISM IN A NONHOMOGENEOUS MICROCOMPUTER CLUSTER

John Meng

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

ASTRACT

Nonhomogeneous microcomputer clusters (clusters containing different types of microcomputers) based on MIDAS-like parallelism and a master-slave hardware control scheme, allow us to parse problems, for parallel execution, in unique ways dependent on the differences in individual microprocessor devices. We describe our microprocessor control scheme and selective developmental applications of the MIDAS parallel architecture to such a cluster. We also briefly describe several specific applications of the cluster.

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

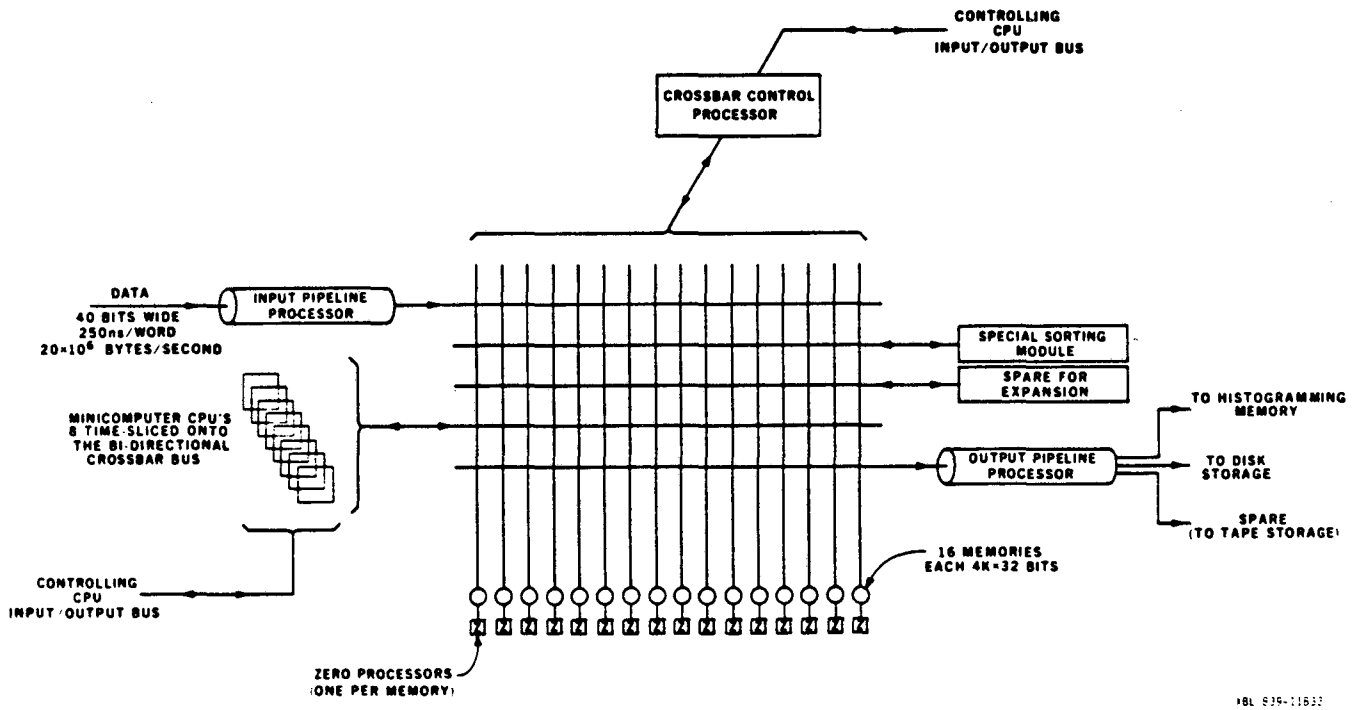


Fig. 1 In the MIDAS architecture, special purpose processors work with a cluster of minicomputer general purpose processors. Data passes through the architecture in memories connected to a preselected sequence of processors. A controlling minicomputer (not shown) is the master of this part of the system.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

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]

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 iterative 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.

CONTROLLING A CLUSTER

Crossbar parameters limit the ultimate size of the MIDAS-like cluster. The number of gating elements in a crossbar configuration increases as the product: (number of processors) times (number of memories). In MIDAS the arithmetic processors, minicomputer Central Processors (CPU's), need memory access infrequently relative to memory speed, so switched memory signals for eight CPU's pass through a single bus under control of a time-slicer. The result is a 6 x 16 structure (96 cross points) instead of 13 x 16 (208 cross points). We are expanding the cluster to 16 arithmetic processors and 24 switched memories. By sending four bits of control through a fully-implemented crossbar, thus bypassing the time-sliced bus during setup and address-sequential memory accesses, we effect a significant bus traffic reduction. By using First In First Out (FIFO) buffers to supply and to collect bus data, we derandomize bus transfers and avoid statistical bunching and the resulting instantaneous slowdowns. These enhancements, plus a change to state-of-the-art faster bus switching devices, stretch the capacity of the bus time slicing scheme to its practical limit. Variables determining relationships among number of processors, bus enhancements and overall performance are too device dependent and too software dependent to accurately predict. We guess, based on MIDAS performance, that 16 CPU's will be adequately serviced by the enhanced bus.

In moving to microprocessor CPU's, we get total control of the chip by intervening between it and its memory and by manipulating its control signals.[8] In MIDAS, where we use minicomputer CPU's, complete control is achieved by manipulating operator control panel lines.[7]

We control a nonhomogeneous microcomputer cluster from the logic represented in Fig. 2. This is adaptable to controlling single chip CPU's as well as Arimetic Logic Units (ALU's) in use as special purpose processors or as pipeline processor units.[8]

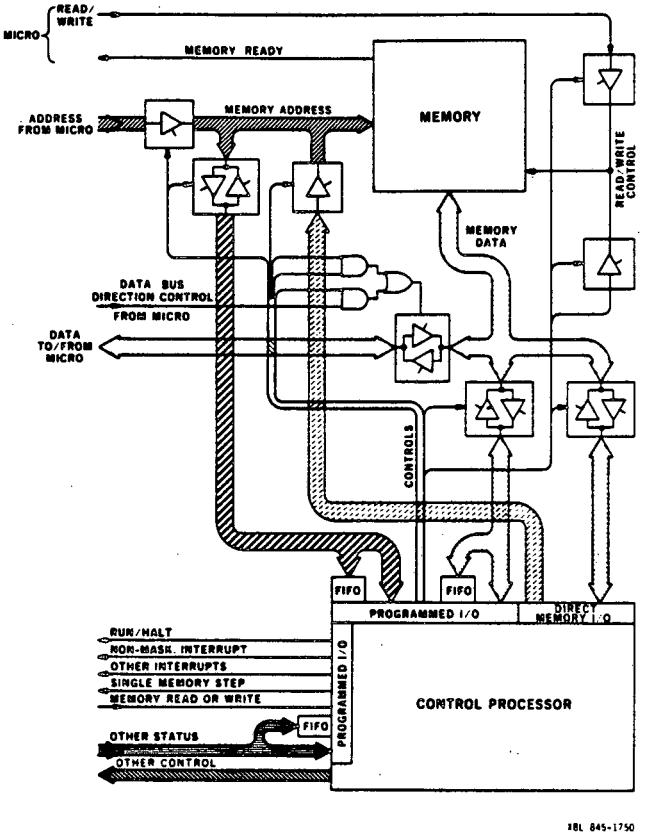


Fig. 2 This is the essence common to all microprocessor connections to the control processor. FIFO connections to high speed buses provide snapshots of bus activity to the control processor.

In MIDAS, we non-intrusively monitor cluster activity through the operator control panel status signals. In microcomputer clusters using the controls of Fig. 2, the attached control computer monitors activity using First In First Out (FIFO) buffers on data buses, address buses and status lines. This handle on cluster element activity and the addition of dynamic reconfigurability to the cluster provide the tools necessary to guarantee optimum functioning within the cluster.

Successfully load balancing the system guarantees that each element is continuously productive, and the ability to dynamically reconfigure the elements within the cluster, applying more or fewer processors depending on use or data patterns, allows us to successfully load balance. In order to reconfigure; that is, in order to change a processor's job during operation, the controlling processor waits for an idle state and downloads new code or restarts at a new location, assuming the cluster contains more than one of each of the different processor types.

EXTENDING THE ARCHITECTURE

With a 16 processor, 24 switchable-memory system we feel we are reaching the upper limit on our ability to afford to implement crossbar-type connections from memory to processor. In order to further extend the system, we propose pyramiding clusters (Fig. 3). Within the pyramid is a top cluster connecting to three clusters one level down. Each cluster connects to three more, etc. Faster expanding pyramids are practical and communicate cluster-to-cluster faster and easier, but too much resource sharing can cause slowdowns from contention.

the structure, circumventing this problem. Our extended architecture depends on shared processors and memories communicating cluster-to-cluster.

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 calls for specialized processor types as well as for general purpose processors.

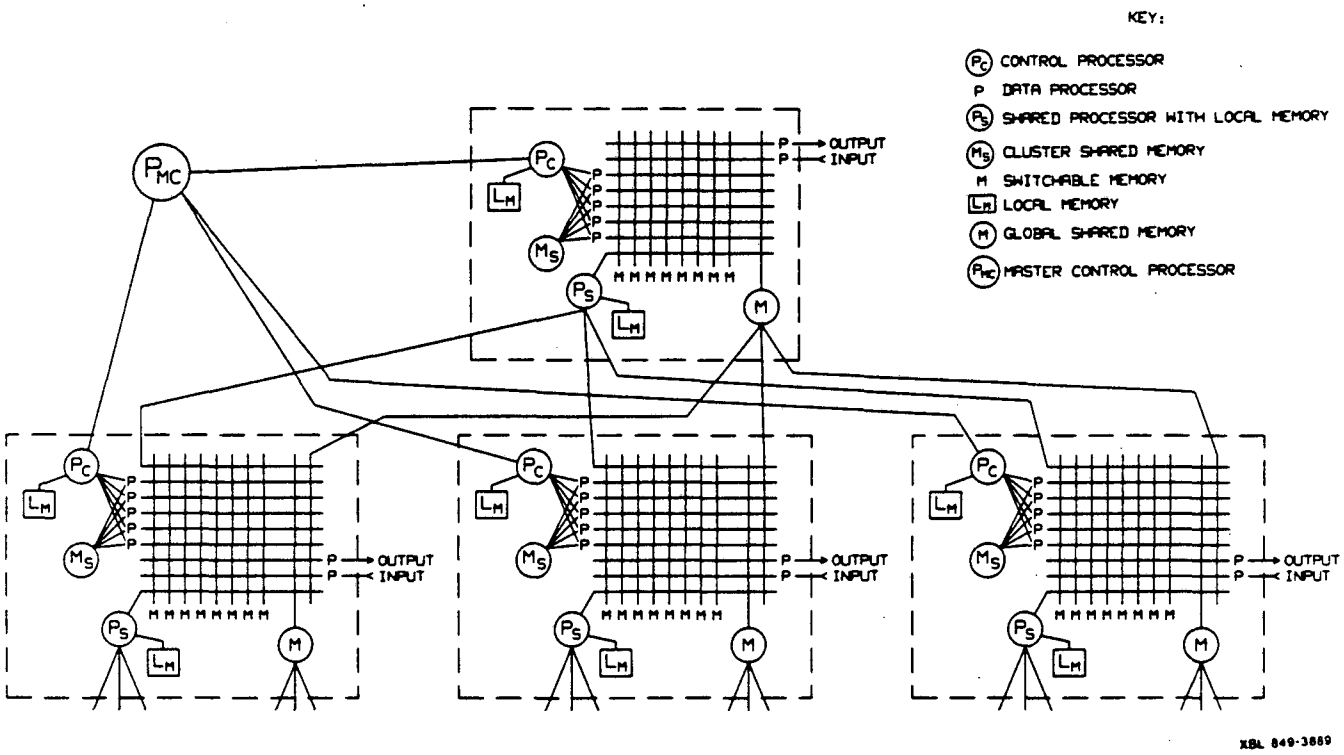


Fig. 3 This is one possible way to extend the MIDAS architecture without letting crossbar dimensions become unwieldly. Cluster levels share both processors and memories. Input and output pass through a dedicated crossbar.

A major bottleneck for many applications is the speed with which data passes into and out of the system. Within MIDAS, as well as in our future systems, each cluster contains one processor handling data flowing into the cluster and one handling data flowing out. We propose a crossbar connection to select data storage devices and/or data sources for use by selected clusters. A master control processor globally controls pyramid control processors, input and output, and information flows to or from the system operator.

In summary, crossbars become unwieldly quickly as processor and memory populations increase. So to extend our architecture, we rationally distribute small crossbars throughout

Available hardware favors three general classes of processor: the general purpose central processor; the special processor dedicated to a single function (either by firmware, hardware or software); and the pipeline processor. Our generic connection (Fig. 2) will connect to each.[8]

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. 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. 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 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 iterative 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.

SOLUTIONS TO SOME GENERAL PROBLEMS

We are applying the MIDAS cluster to solve a variety of specific problems.[2,5,6,9] Based on this experience and its extrapolation to newer cluster architectural enhancements and multiple clusters, the following applications seem particularly viable.

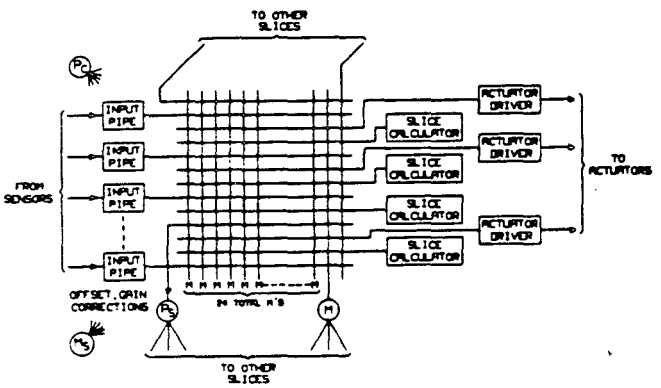


Fig. 4 The figure control processor is a functional link in a complex servo loop, and in three-dimensional systems may involve multiple clusters, each of which controls one slice. Pipeline processors normalize sensor inputs and memory movement sequences collect appropriate control data for each actuator.

The mechanical figure control system (Fig. 4) 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.

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,[9] submatrices may be used in the calculations, and the results combined. The same may be true with matrices derived from CCD imaging systems. If CCD images are being analyzed, a Fast Fourier Transform (FFT) processor or a correlative template operations processor might be useful to the cluster.

Tree structure approaches to solving problems make use of a dedicated processor to compile statistics on frequency of use of pathways through the tree, 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. 5). Common examples are games, such as Othello and Chess, and knowledge-based systems.

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 direc-

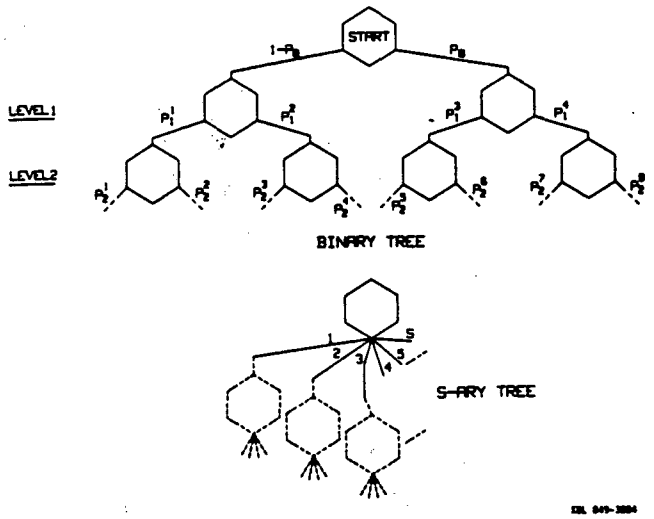


Fig. 5 In tree structures moving down to the next level requires knowing the path from the previous level. Consequently, applying parallelism to the tree structure depends on making educated guesses of the exit path based on probabilities P_n, then calculating as far ahead in depth and in breadth as parallel resources permit. Speedup is a function of the success of our guessing strategy and ultimately of the effectiveness of applying appropriate feedback to values of P_n.

tions and depths. Figure 6 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.

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. 2) and pass them to cluster control processors for passage on demand to cluster processors. Other distributed schemes have been used.[10] Decipher run results are a simple yes-no answer to each key, and only the yes need be

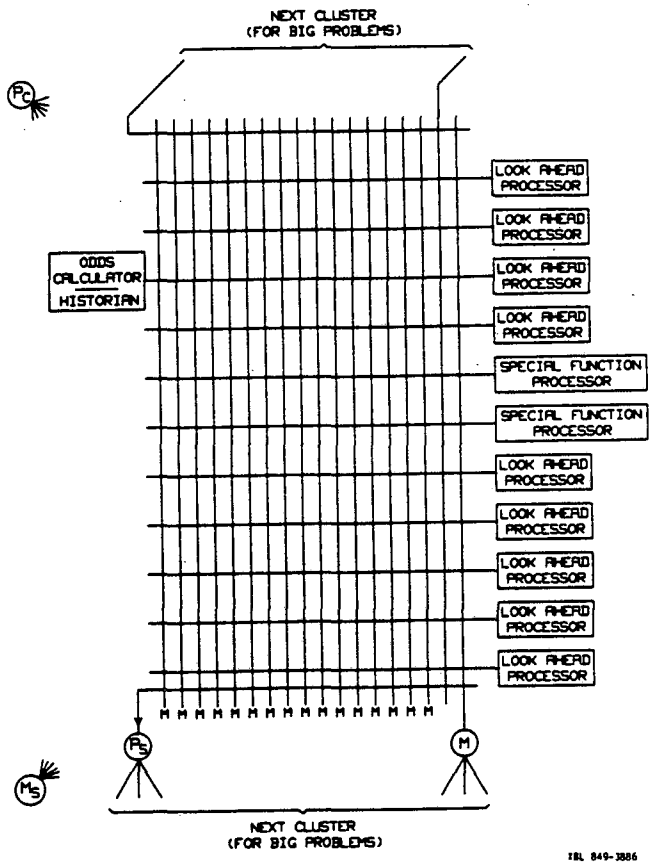


Fig. 6 Problem solutions paralleling the tree structure of Fig. 5 yield to this cluster structure. The number of look-ahead processors, in conjunction with the strategy cleverness of the odds calculator, determines the efficiency of parallel processor application to the the type of solution.

explicitly communicated back to the master control processor. Monte Carlo calculations result in an output for each random input. These accumulate in switchable memories, m, until full, when they are switched to the data storage processor for emptying. Alternatively, they are switched to a globally shared processor and collected in a cluster dedicated to data accumulation and user interaction, allowing the user to view the emerging statistically generated results.

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 speedups.

SUMMARY AND CONCLUSIONS

An architecture based on multiple processors attachable to memory modules through a crossbar interconnect is under investigation in the MIDAS

prototype cluster. The future of this architecture seems to be with multiple clusters of arrays of different types of microprocessors working in concert under the direction of cluster control processors which, in turn, are controlled by a single master control processor. By expanding into multple clusters, we avoid the problem of square-law crossbar gate count increases which would occur with single cluster expansion.

Preliminary explorations, in many cases with the aid of the MIDAS cluster, lead us to believe the cluster architecture, logically expanded into a multi-cluster architecture will realize the nearly linear increases in processing speed observed within the MIDAS single cluster. Scenarios have been developed and in several cases are in use applying this structure to the solution of a diversity of 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, "Controlling a Radially Connected Array of Minicomputers", Conf. Record of the Sixteenth Asilomar Conf. on Circuits, Systems and Computer, 280 (1982).

8. 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).

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

10. D Logan, W Rathbun and C Maples, "Parallel Indpendent 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