COMPLETING AN MIMD MULTIPROCESSOR TAXONOMY

ERIC E. JOHNSON

ABSTRACT

MIMD multiprocessor architectures have been classified as shared memory, message passing, or "hybrid" architectures. This taxonomy is shown to be incomplete, and an alternative complete taxonomy is suggested. Examples of each class of the taxonomy are discussed, along with general attributes of the classes.

INTRODUCTION

Flynn's taxonomy of computer systems[1] has proved to be a useful tool for differentiating among uniprocessors (SISD), array and systolic machines (SIMD), and general multiprocessors (MIMD). With the current activity in the multiprocessor area, an informal taxonomy of MIMD systems has developed; examples of this taxonomy may be found in recent papers on programming MIMD machines[2,3].

In this taxonomy, machines which employ a shared global memory for interprocess communication and synchronization are termed **shared memory** machines. Architectures such as the hypercube in which the system's memory is distributed among the processing elements, and which employ interprocess messages for communication and synchronization are termed **message passing** architectures. Karp[2] noted that machines such as the BBN Butterfly[4], in which memory is distributed but communication and synchronization take place through shared variables, form a class which is distinct from shared memory machines; he termed this class **hybrid** architectures. Howe[3], on the other hand, classified all machines employing the shared variable programming model, including the Butterfly, as shared memory machines; this suppresses some significant differences in programming and performance which arise between global and distributed memory architectures.

In a global memory machine, the memory modules are equally accessible from any processor, and need not exist in a 1:1 correspondence with the processors; distributed memory architectures, on the other hand, usually associate one memory module with each processor, and grant each processor special access to this associated memory module. Placement of code and data in a distributed memory machine must take into account this non-uniform accessibility of memory to processors in order to avoid the performance penalty of non-local accesses; as noted by Lundstrom[6], however, the primary concern in global memory machines is to place code and data to minimize contention for specific memory modules.

Thus, we find that both the communication/synchronization mechanism and the memory structure employed in an architecture may have significant effects on the performance, and on the programmer's view of that architecture. The topology of the processor-memory interconnections (e.g., bus, crossbar, multi-stage switched network, etc.) also has an impact on the performance of the system, but appears to have only second-order effects on the programmer's model.

PROPOSED TAXONOMY

The architectural features of MIMD machines which are employed in this classification scheme are memory structure (global vs. distributed) and communication/synchronization mechanism (shared variables vs. message passing). The resulting MIMD taxonomy contains the four classes shown in Figure 1:

1) Global Memory-Shared Variables (**GMSV**) -- the shared memory machines,
2) Distributed Memory-Shared Variables (**DMSV**) -- the hybrid machines,
3) Distributed Memory-Message Passing (**DMMP**) -- the message passing machines,
4) and Global Memory-Message Passing (**GMMP**) machines.

Subclasses differentiated by interconnection topology may be designated as, for example, GMSV(bus) for a bus-based global memory shared variable machine.

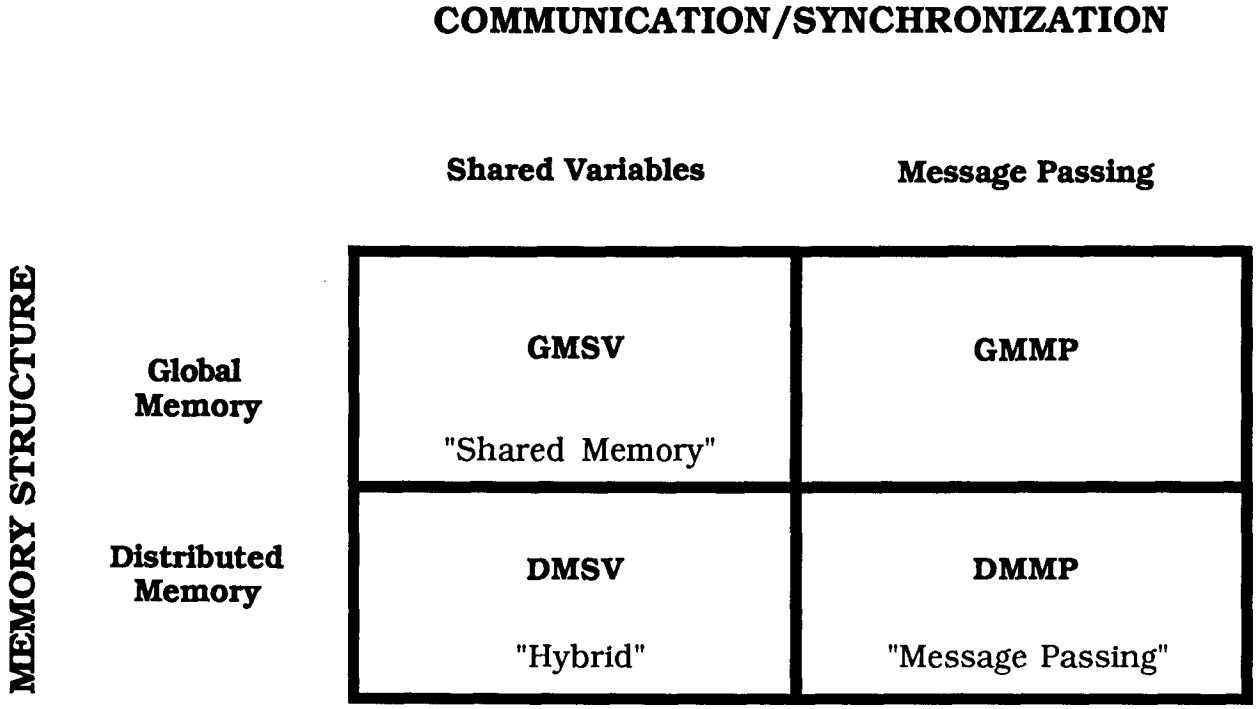

Figure 1: A Taxonomy of MIMD Parallel Architectures

DISCUSSION

It is apparent from Figure 1 that the current informal taxonomy is incomplete, in that it omits the GMMP architectures. Although GMMP machines are not as common in the literature as GMSV and DMMP machines, they are not completely unknown: two examples are the ELXSI 6400[9] and the Virtual Port Memory research machine under construction at New Mexico State University[10].

This taxonomy appears useful for discussing the characteristics of the current crop of MIMD multiprocessors. For example, GMSV machines share the advantages and disadvantages of the shared variable programming model (e.g., the requirement for synchronization of accesses to shared data) and the uniform accessibility of a global memory from any processor. Memory contention is a central issue in the design of a GMSV machine; consequently, such architectures may be compared by examining the approaches used to manage or reduce this contention. Two techniques commonly employed in machines larger than 12 or so processors (a realistic limit for simple bus-based GMSV machines[3]) are the use of advanced processor-memory interconnections, such as the omega network in the NYU Ultracomputer[11], and the use of cache memory to reduce the rate of accesses to the global memory, although the shared variable programming model used in GMSV machines requires the use of a cache coherency protocol.

DMSV machines, such as the Butterfly[4], share the shared variable programming requirements of synchronization and cache coherence protocols with the GMSV machines, but are distinct from GMSV architectures in their requirements for correct code and data placement for maximum performance.

DMMP architectures include both hypercubes, such as the NCUBE[12] or FPS T-Series[13] machines, and mesh-connected machines such as the AMETEK 2010. The isolated process address spaces provided by such architectures apparently simplify the task of debugging the parallel programs they are to execute[2] and allow the use of private cache memories without cache coherence problems, but, as for DMSV machines, the programmer of DMMP machines must deal with the data placement problem to achieve full performance.

Finally, machines of the GMMP class have isolated process address spaces (usually virtual address spaces) with uniform access to memory, and may employ private caches to reduce the rate of accesses to the global memory without the need for cache coherence protocols. If a GMMP architecture provides some mechanism to reduce the performance penalty of data movement due to the message passing semantics, it should provide an attractive combination of features.

PERFORMANCE CHARACTERISTICS

An interesting characteristic of a multiprocessor architecture is the effect on its performance of increasing the degree of process coupling. For architectures of the shared variable classes (GMSV and DMSV), we may measure process coupling by the fraction of memory references made by an algorithm which update shared variables. For GMMP and DMMP machines, an equivalent measure is the ratio of interprocess communication to instruction execution.

Defining this process coupling parameter as R, we would expect, in general, that a machine will perform quite well when R is small because the processors have minimal impact on each other. As R increases, contention for global memory, processor-memory access paths, or interprocessor communication links increases, and the overall performance of the system drops. Because memory access times seen by a processor set a bound on processor performance, and because average memory access time is a parameter fairly easy to determine by simulation or analysis, we can find a crude performance curve for an architecture in its average memory access time versus R.

We have employed this technique in an exploration of the four classes of the proposed taxonomy. Instances of the four MIMD architectural classes have been analyzed[5] and simulated over a range of values for R. As expected, all four machines experienced little degradation in memory access times for small values of R (0.001). As R increased to 10%, average memory access times for the GMSV and DMMP machines doubled, that for the DMSV machine increased tenfold, but the GMMP average memory access time increased by only a few percent. The results correlate well with experiences

reported by independent investigators[4,6,7,8]. Research is continuing in this area, and a full discussion of the models and results mentioned above will be presented in a future paper.

CONCLUSIONS

The informal MIMD taxonomy of "shared memory," "message passing," and (occasionally) "hybrid" classes of multiprocessors should be replaced with a formal set of classes, such as GMSV, GMMP, DMSV, and DMMP, which was suggested here. Such an organization should provide a useful framework for discussion of architectural issues, and for reporting analysis, simulation, and measurement results.

REFERENCES

1. M.J. Flynn, "Very High-Speed Computing Systems," *Proceedings of the IEEE*, vol. 54, no. 12, December 1966, pp. 1901-1909.

2. A. Karp, "Programming for Parallelism," *Computer*, vol. 20, no. 5, May 1987, pp.43-57.

3. C. D. Howe and B. Moxon, "How to Program Parallel Computers," *IEEE Spectrum*, vol. 24, no. 9, September 1987, pp. 36-41.

4. W. Crowther *et al.*, "Performance Measurements on a 128-Node Butterfly Parallel Processor," *Proceedings of the 1985 International Conference on Parallel Processing*, pp.531-535, 1985.

5. E. E. Johnson, "The Virtual Port Memory Multiprocessor Architecture," PhD Dissertation, New Mexico State University, 1987.

6. S. F. Lundstrom, "Applications Considerations in the System Design of Highly Concurrent Multiprocessors," *IEEE Transactions on Computers*, vol. C-36, no. 11, November 1987, pp.1292-1309.

7. J. L. Gustafson, G. R. Montry, and R. E. Benner, "Development of Parallel Methods for a 1024-Processor Hypercube," to appear in *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 4, July 1988.

8. J. Sanguinetti, "Performance of a Message-Based Multiprocessor," *Computer*, vol. 19, no. 9, September 1986, pp.47-55.

9. R. Olson, "Parallel Processing in a Message-Based Operating System," *IEEE Software*, July 1985, pp. 39-49.

10. E. E. Johnson, "A Prototype Virtual Port Memory Multiprocessor," Technical Report NMSU-ECE-88-003, New Mexico State University, Las Cruces, 1988.

11. A. Gottlieb *et al.*, "The NYU Ultracomputer — Designing an MIMD Shared Memory Parallel Computer," *IEEE Transactions on Computers*, vol. C-32, no. 2, February 1983, pp. 16-24.

12. *NCUBE User's Manual*, NCUBE Corporation, Beaverton, Oregon, 1987.

13. S. Hawkinson, *The FPS T-Series: A Parallel Vector Supercomputer*, Floating Point Systems, Inc., Beaverton, Oregon, 1986.