RECENT ADVANCES IN PARALLEL SCIENTIFIC COMPUTATION IN IBM

DAVID B. SOLL
IBM Corporation
High Performance Computing Solutions Development
Kingston, New York

ABSTRACT

IBM research programs and joint IBM-University study efforts have resulted in advances in parallel scientific computation which have contributed to the state of the art. Some of these, as they mature, have been included in IBM products and offerings. This paper describes some of these hardware and software activities and present some results.

I. *INTRODUCTION*

During the past several years, a number of research projects at IBM and joint IBM-university study efforts have been conducted into various aspects of parallel processing, particularly as it pertains to scientific computation. These include the experimental processors, RP3, GF11 and VULCAN, experiments with the IBM 3090 vector multiprocessor, the 3090 as a host to multiple attached processors, and clusters of 3090 multiprocessor complexes. Investigations into parallel software have included the development of Parallel FORTRAN, Clustered FORTRAN, and the LCAP/3090 system. This paper presents an overview of these studies along with some representative results and conclusions.

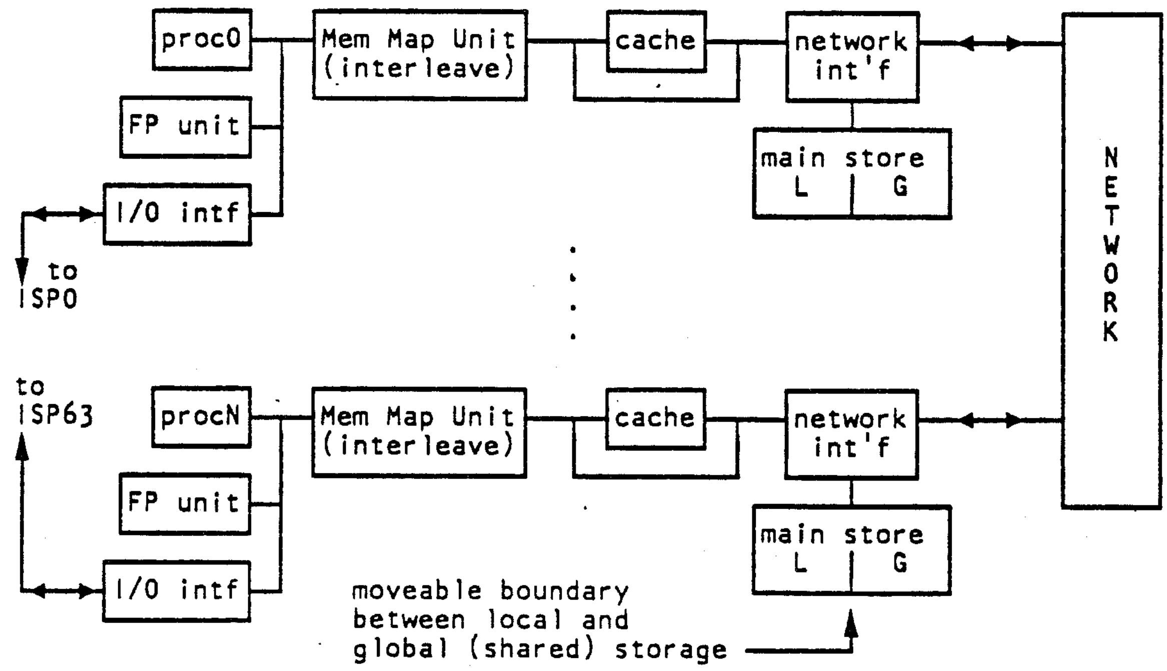
II. *PARALLEL PROCESSORS*

RP3

The Research Parallel Processing Prototype (RP3) was initiated in the IBM Research Division, in cooperation with the Ultra Computer project of the Courant Institute of New York University. The RP3 machine is a highly parallel MIMD design using both a shared memory paradigm (after the NYU Ultracomputer) and a local memory, message-passing paradigm (after the Cal-Tech Cosmic Cube), as well as mixtures of the two., chosen at run-time. The objective of the research effort was to investigate both hardware and software aspects of highly parallel computation.

The RP3 was designed to accommodate 512 processing nodes, each with a 32 bit microprocessor (ROMP) plus vector floating point hardware, with a peak of 1.3 GIPS (800 MFLOPs). The research was actually carried out on a one-octant (64-way) configuration. The interconnection network is actually composed of two networks. The first is a low-latency network consisting of a rectangular SW banyon, similar to an Omega network, but providing dual source-sink paths. The second interconnection is a combining network with the geometry of Lawrie's Omega network. This network has the ability to combine messages directed to the same memory location. More complete details of the architecture and design of RP3 may be found in several papers included in ref 1. A diagram of the data flow of the RP3 system is shown in figure 1.

Figure 1: RP3 System Data Flow



Several interesting results from this research have been obtained and reported in the literature. The often proposed practice of access to shared memory on a highly parallel multiprocessor system is to utilize a message-or packet-switched multistage switching network. An initial results of the RP3 project was that a type of network traffic non-uniformity, a "hot-spot", can produce effects that severely degrade ALL network traffic, not just the traffic to the shared locks which produced the "hot-spot". This effect was called "tree saturation", and was determined to be quite general; that is, it is independent of network topology, switching mode (packet or circuit) or whether the network is used for message passing or memory access. It is a characteristic of a multistage network with distributed routing whose traffic pattern exhibits a "hot-spot" non-uniformity for any reason. A further result was that message combining, originally proposed to solve a different problem, is an effective technique for dealing with this problem when is arises due to global shared locks. This result was extended to the observation that it is possible to avoid the "hot-spot" contention with single applications specifically tailored to the connection topology, or through the use of combining for some cases of interest, but avoidance is infeasible in multipurpose, generally used systems.

GF11

The GF11 parallel computer is a single instruction, multiple data (SIMD) processor, developed and constructed at IBM T. J. Watson Research Center, and has been operational since October 1989. It consists of 566 identical processing elements, each capable of 20 MFLOPs operation, connected via a communication network which allows the processors to dynamically reconfigure themselves into arrays of various dimensions and sizes, or other interesting interconnection patterns such as tree or cube. The network provides 11.3 GBytes/sec bandwidth, and reconfiguration can take place on every word transfer without sacrificing that bandwidth. The network is a three stage (576x576) Benes network, organized as three stages of 24x24 crossbar switches. Overviews of the GF11 machine organization and the GF11 processor are shown in figures 2 and 3. Descriptions of the GF11 design and implementation may be found in references 2 and 3.

The GF11 was originally designed for numerical verification of the predictions of Quantum Chromodynamics (QCD). The formulation of QCD, approximated by a hypercube lattice has been mapped onto 512 of the GF11 processors, resulting in a sustained rate of 9.5 GFLOPs, while a conjugate gradient algorithm implemented on 500 active processors yielded 8.5 GFLOPs. Depending on the choice of lattice size and the requested permutation of the GF11 processors, the peak rates are in the 8.5 to 10.1 GFLOPs range.

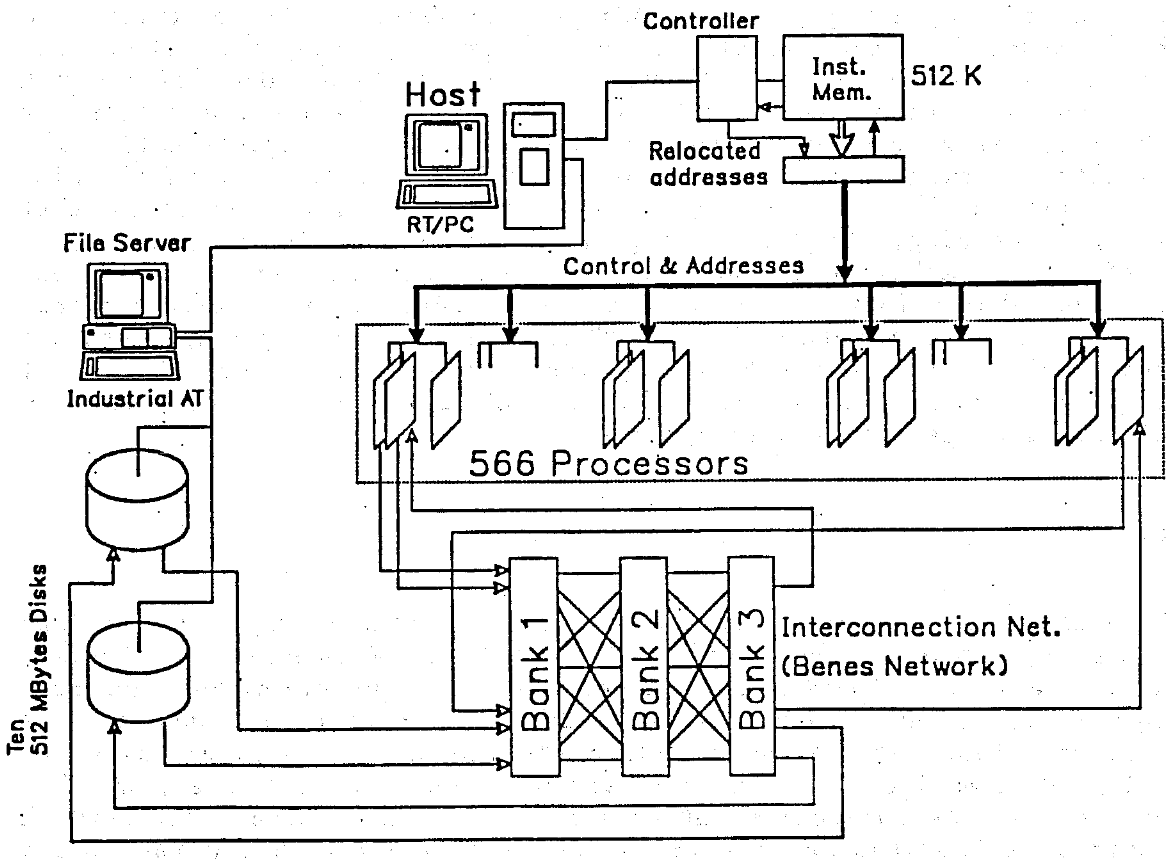


Figure 2: Overview of GF11 Machine Organization

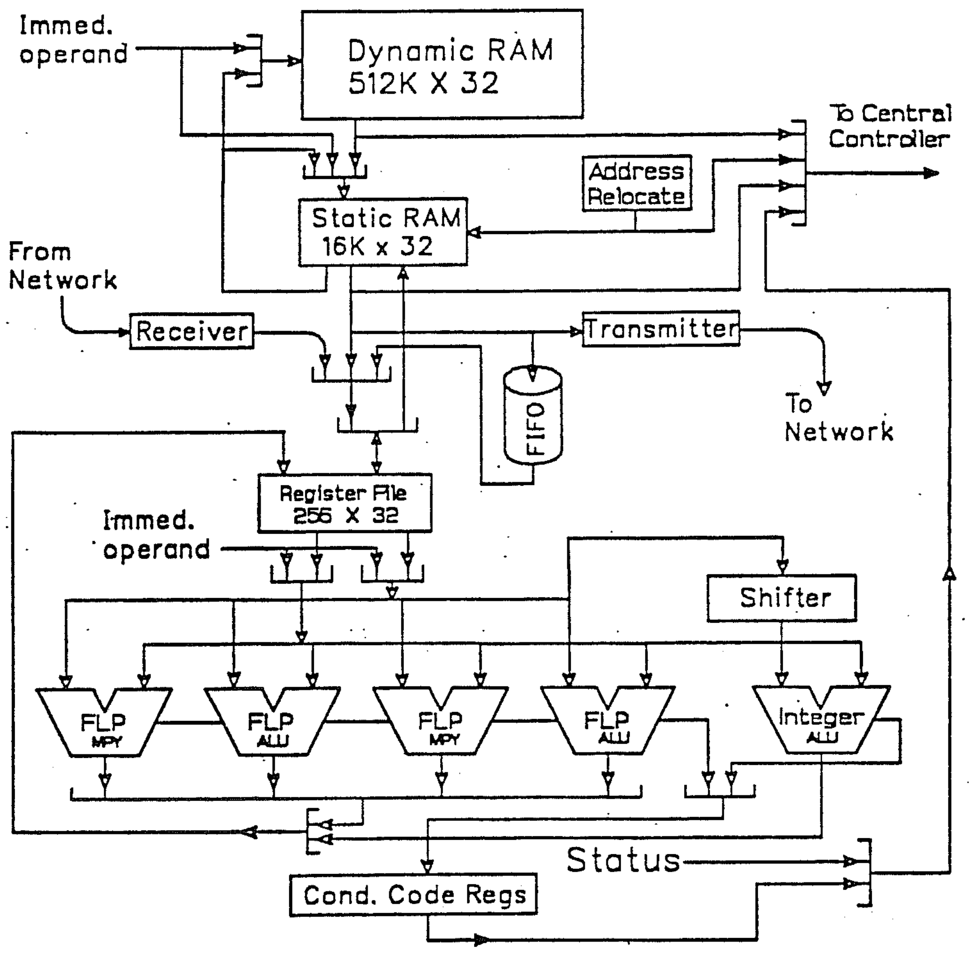


Figure 3: Overview of a GF11 Processor

Other applications which have been implemented on the GF11 include the Shallow Water equations, Matrix Multiplication, a simulation of galaxy evolution and backpropagation learning. The Shallow Water benchmark computes the time dependent solution of the equations on a 256x256 rectangular grid. At the time this problem was implemented, only 300 of the processors were operational, and 256 of these were used. Two structures of the data were implemented; a partitioning of the grid into 16x16 squares, which minimizes interprocessor communication did not provide any advantage over the 256x1 partitioning which maximizes communication but has a simpler code structure. This is performance is primarily due to an imbalance between the number of ADD and MULTIPLY operations. This application achieved 81% efficiency, which translates to approximately 8.2 GFLOPs for a full machine configuration.

Matrix Multiplication, also implemented on 256 processors, used a blocked systolic algorithm for multiplying two 1024x1024 matrices. This computation achieved 99% of peak, which translates to 5.1 GFLOPs for 256 processors.

The Galaxy code (3) is a time-dependent, three dimensional hydrodynamic model of a system of stars and gas which simulates the dynamics of a representative piece of interstellar matter. This code does not map closely onto the GF11. The computationally intensive parts were divided into nineteen sections, with another ten for other, non-essential purposes. Each section of the code has its own local operational balance which prevents complete utilization of GF11. This code runs with an efficiency of close to 70%, resulting in a speed of 3.4 GFLOPs on a 256 processor GF11.

The backpropagation learning algorithm is a neural network simulator, (ref 4) applied to the problem of speaker-independent continuous speech recognition. Although generally considered to be best suited to an MIMD configuration with very fast communication, the application may also be efficiently executed on an SIMD using the technique of having each processor run the same network, if the per-processor memory is sufficient. Two different connections were employed in this investigation; ring and tree. The relative performance of these two choices, using the NETTALK (4) text-to-phoneme benchmark is shown in table 1 from reference 4 In terms of a metric which is specific to this application, i.e. millions of connections per second (MCPS), it is clear that the tree connection is the better choice. A comparison of other implementations of backpropagation learning on other machines is shown in table 2 from ref. 4. The figure of 901 MCPS on 256 is equivalent to 3.8 GFLOPs. as a theoretical limit, for very large data, the authors of ref. 4 have estimated a value of 1900 MCPS, or 8 GFLOPs.

TABLE 1: MILLIONS OF CONNECTIONS PER SECOND, RING AND TREE

MILLIONS OF CONNECTIONS PER SECOND

PROCESSORS	TREE MCPS	RING MCPS
8	26	26
16	55	53
32	112	107
64	216	170
128	415	222
256	753	180
356	901	-
512	1231	84

TABLE 2: COMPARISON OF LEARNING SPEEDS FOR VARIOUS MACHINES

MILLIONS OF CONNECTIONS PER SECOND

MACHINE	MCPS
VAX	0.008
SUN 3/75	0.01
VAX 780	0.027
SUN3/160/FPA	0.034
RIDGE 32	0.05
DEC 8600	0.06
CONVEX C-1	1.8
16K CM-1	2.6
CRAY-2	7
64K CM	13
WARP	20
CM-2	40
GF11	901

LCAP

The Loosely Coupled Array of Processors or LCAP System was conceived with a clearly defined set of applications in mind (ref. 5). These applications consist of large-scale applications in Theoretical Chemistry. The class of applications was broad enough so that LCAP would not be considered as a special purpose configuration, but still could not be classified as general-purpose either. In fact, the objective was to use parallel processing to solve real problems, rather than to conduct experimentation into parallel processing. LCAP is operational at the IBM Center for Scientific & Engineering Computation in the High Performance Computing Solutions Development organization in the Kingston, New York laboratory.

Nevertheless, the evolution of the LCAP system has resulted in its use for a significantly broad spectrum of applications, so that characterizations of LCAP with respect to general principles of parallel processing may be made that apply to other MIMD machine organizations. It is appropriate to discuss some of the applications which led to the design of LCAP and their influence on LCAP/3090.

These applications primarily consisted of large problems in Theoretical Chemistry. This area includes Quantum Chemistry using Self-consistent Field Theory, and Statistical Mechanics using Monte Carlo and Molecular Dynamics methods. The applications share the following important properties: first, the parallelism in these calculations is obvious. They employ computations over ensembles of particles acting in combinations (2,3,...). Second, each n-body computation is quite complex, requiring a large amount of computation (and time) for each, growing as the number of interactions grows. This application type is generally considered as having "very-large-grained" parallelism. This also implies that there is a great deal of computation prior to the need for any communication between processors. The third attribute is that only a small part of the computation involves global simulation, and thus is very well suited even to a distributed collection of processors.

The first system, LCAP1, consisted of an IBM 4381 host, with 10 Floating Point Systems FPS-164 attached processors (AP's), using IBM's Virtual Machine (VM/SP) operating system. A second, attached system consisting of an IBM 4341 and 3 AP's was also configured and used for algorithm and program development. These configurations are shown in figure 4, with IBM channels for communication and other associated switching and disk storage. At the same time, a second, more powerful version, LCAP2 was assembled. This consisted of an IBM 3081 processor as host, with 10 FPS-264 AP's attached, and used the IBM MVS/SP operating system. Representative performance is shown in tables 3 and 4 (ref. 5). It should be noted that although these applications were not optimized for their respective configurations, the measurements serve to indicate the aggregate computational power of the systems.

Applications performance on ICAP/1

Job	Elapsed Time for ICAP/1 (minutes)				
	1 AP	3 APs	6 APs	10 APs	CRAY-XMP
Integrals (27 atoms)	71.7	24.0	12.3	7.8	7.6
SCF (27 atoms)	25.2	9.4	5.9	4.9	3.6
Integrals (42 atoms)	203.7	68.9	38.3	21.2	23.2
SCF (42 atoms)	73.0	26.0	14.3	10.6	8.7
Monte Carlo	162.1	57.8	32.0	22.0	20.4
Molecular dynamics	99.6	34.6	19.3	13.7	17.0
Seismic	33.8	11.8	6.6	4.3	5.6

Table 3

Applications performance on ICAP/2

Job	Elapsed Time for ICAP/2 (minutes)				
	1 AP	3 APs	6 APs	10 APs	CRAY-XMP
Integrals (27 atoms)	19.1	6.5	3.3	2.3	7.6
SCF (27 atoms)	10.6	5.2	3.7	3.4	3.6
Integrals (42 atoms)	55.0	18.7	9.3	6.1	23.2
SCF (42 atoms)	24.1	9.1	5.6	4.7	8.7
Monte Carlo	60.0	20.9	11.4	7.7	20.4
Molecular dynamics	29.6	10.6	5.9	4.2	17.0

Table 4

Since the LCAP program included many collaborative participants from universities and other research institutions, it is possible to demonstrate parallel speed-up for a number of diverse application areas. These results, taken from reference 5 are given by the ratio of the sequential, single processor time to the time to run on N processors in figure 5. However, some engineering applications, using standard discretization schemes such as Finite Differencing or Finite Elements did not perform as well due to the grain size of the parallelism. The interprocessor communication burden was too large for the 3 MB/sec. channel speeds employed. A subsequent extension to the LCAP system included bulk, shared memories and a fast bus, developed by Prof. Martin Schultz of Yale University and Scientific Computing Associates attached to the AP's. This configuration is shown in figure 6.

These evolutionary changes eventually led to the LCAP/3090 system, consisting of 4-IBM 3090 model 400 systems (a total of 16 vector processors), coupled by IBM channels. As part of the experimentation many application areas were successfully parallelized on LCAP/3090, including Molecular Dynamics, Fluid Dynamics, Micro-Hydrodynamics, Monte Carlo, Circuit Analysis, Protein Structures, High Energy Physics, Neutron Transport, Atmospheric Studies (pollution migration), Seismic Migration and Oceanography (current flow).A number of parallel algorithms in general use were also successfully implemented. The LCAP effort also included the development of parallel software to preprocess, schedule and manage the parallel jobs. This will be discussed later in this paper.

Figure 4: LCAP1 - Loosely-Coupled Array of Processors

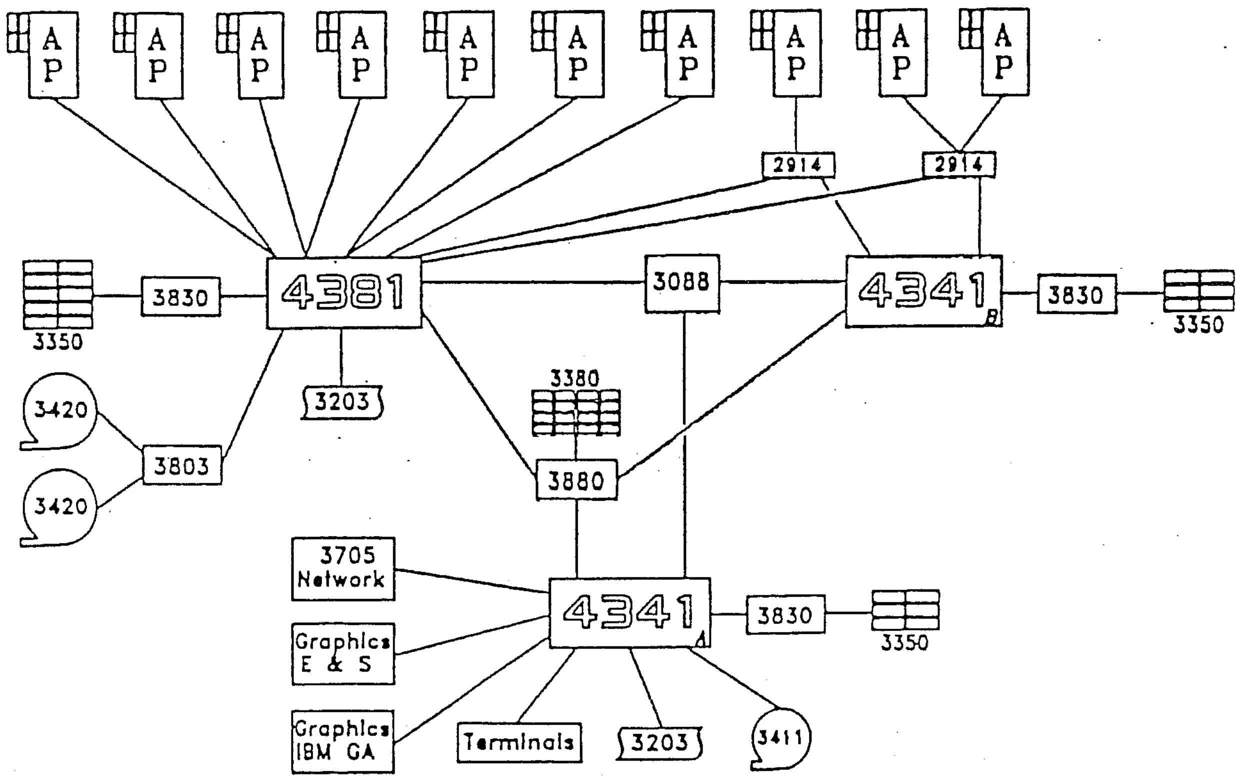


Figure 5: Speedup for Parallel Applications

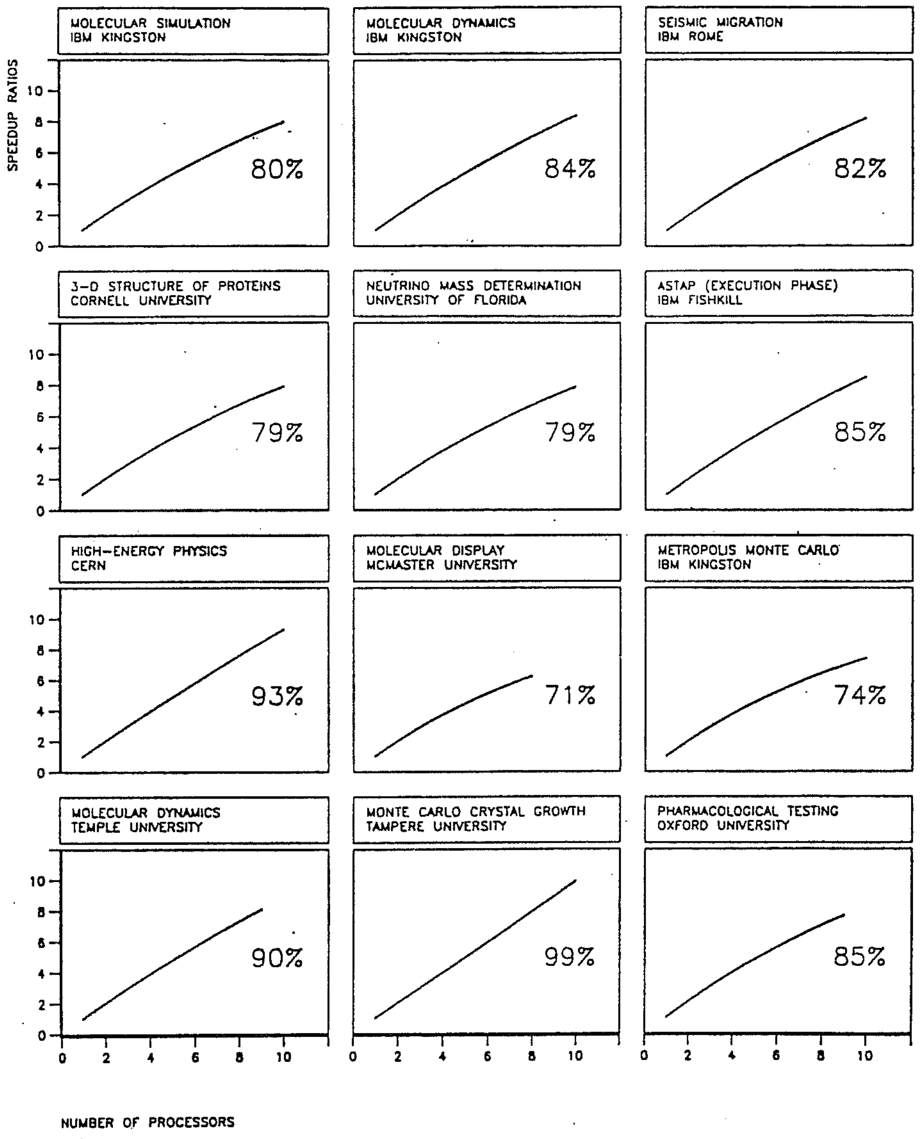
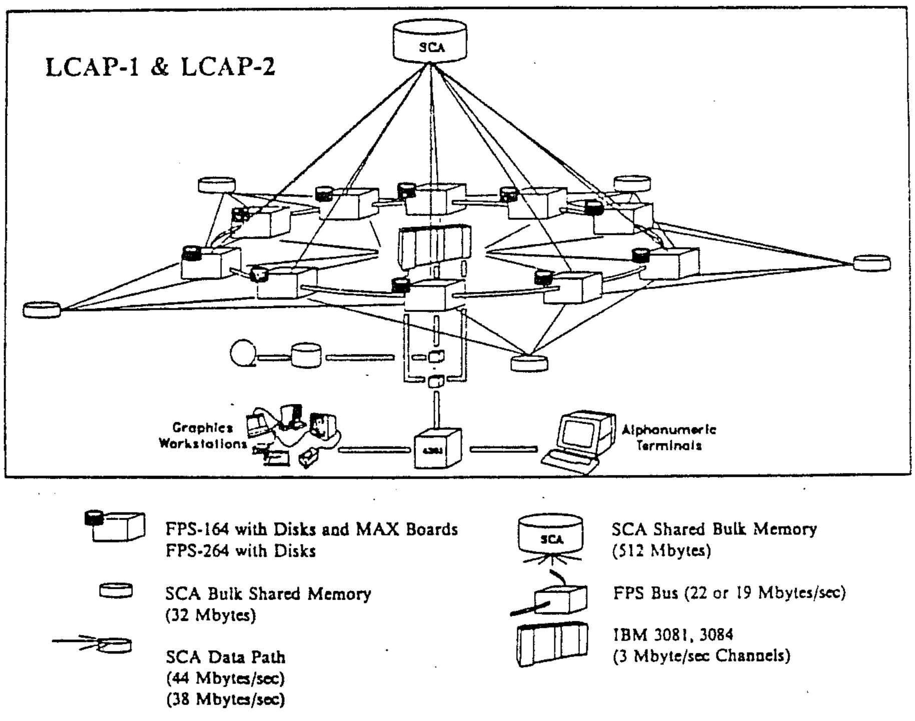


Figure 6: Extended LCAP1 and LCAP2



VULCAN

The VULCAN Supercomputer is a massively parallel MIMD engine being designed at the IBM Research Division. The design of VULCAN is modular, expandable, and is specifically tailored to permit scaling to the teraflop range in performance. The design philosophy is to utilize state-of-the-art, high performance, single-chip microprocessors, with matching memory, communication and DASD storage. Each processor is connected to the switching network by its own high performance (50 Mbyte/sec) channel, using message-passing with many path choices. Utilization of the IBM 4-Mbit memory chip technology permits 32 Mbytes of memory per processor.

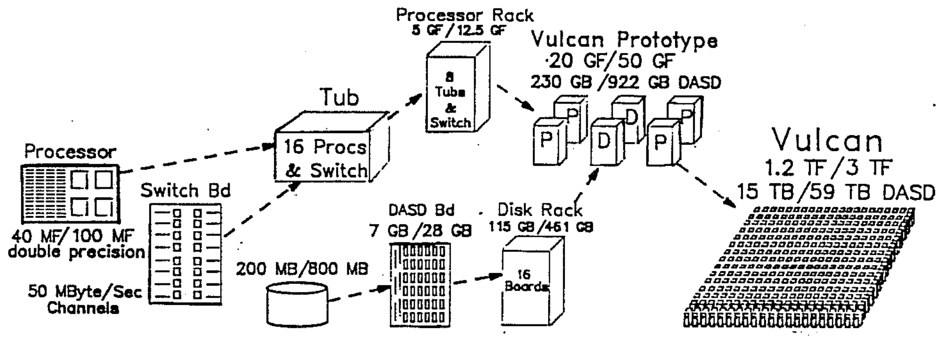
Processors are packaged 4 per board, with associated memory, and 4 boards are grouped to an assembly called a Tub. A Tub consists of 16 processors plus a switch board. (see figure 7), while 8 Tubs comprise one rack (128 processors and 20 switch boards). With a 40 MFLOPs per processor performance, a rack is capable of 5 GFLOPs performance. DASD at 200 Mbyte each is packaged 36 per DASD board, or 7.2 Gbytes, with 16 boards per rack for a total of 115 Gbytes.

The initial prototype, scheduled for operation in the first quarter 1992, will consist of four processor racks and two DASD racks. The 512 processors have a total of 20 GFLOPs peak, 16 Gbytes of memory, and 230 Gbytes of DASD. The total bandwidth at 40 Mbytes/second per processor is 20.5 Gbytes/second, yielding byte-per-flop ratio of 1! This high data delivery rate, or low communication latency implies an opportunity beyond the peak processing speed alone, since it increases the spectrum of potentially effective applications significantly.

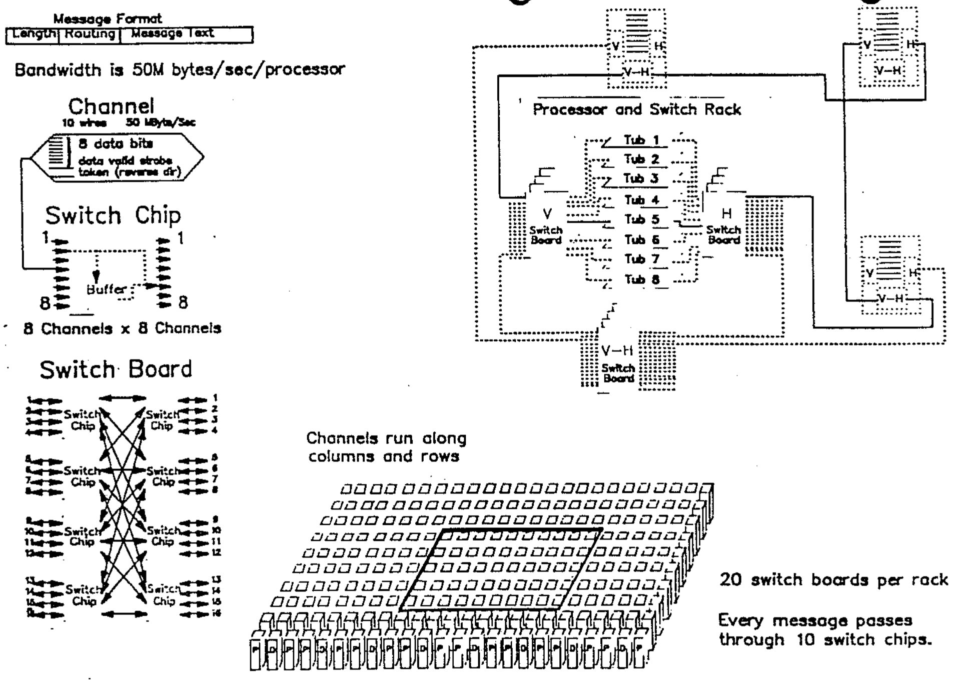
The full VULCAN configuration, consisting of 32,768 processors would scale to 1.2 TFLOPs, 1 Tbyte of memory and 15 Tbytes of DASD while maintaining a constant data rate of 1 byte/flop.

Figure 7: VULCAN Architecture

Vulcan
Machine for Teraflop Performance



Vulcan Message Switching



The IBM 3090 processor complex is, in itself, a tightly-coupled, shared memory, multiprocessor consisting of up to 6 powerful vector (SIMD) processors. Previous papers presented in the 1984 and 1986 workshops on parallel processing at ECMWF (ref 6,7). In those papers, only coarse-grained parallelism using the IBM VS FORTRAN Multitasking Facility was discussed. Subsequent experiments were conducted regarding the limits of the IBM 3090 system used for fine-grained (micro-tasking) parallelism (ref. 8).

At the lowest level, so called "native" microtasking, assembler level macro calls are used to implement parallelism. The basic synchronization overhead was measured as 2 microseconds. In a FORTRAN environment, this overhead, including the cost of subroutine start-up, was shown to be of the order of 5 microseconds for vectorized, micro-scale parallelism. This corresponds to the equivalent floating point synchronization overhead parameter, S 1/2 = 150 (after Hockney & Jesshope, ref. 9). When the effect of vector pipeline start-up is removed to isolate the synchronization overhead alone, this reduces to S 1/2* = 100. This means that effective, low overhead computations of the order of one vector section size (vector register length, 128) on early 3090 models, or one-half of the section size (256) on later models.

At the other extreme, the LCAP/3090 configuration and software, and other efforts described elsewhere in this paper, have been oriented to the use of clusters of 3090 multiprocessor configurations. One such effort includes the coupling of two 3090 model 600 processor complexes at the Cornell University NSF center. The collaborative effort between IBM and Cornell began in 1985 as part of the NSF center's concentration on parallel processing. The initial effort centered around a single 3090/600 (6-processor) system and was central to the development of Parallel FORTRAN, described later in this paper. The subsequent installation and coupling of a second 3090/600 system provided the opportunity to investigate a loosely-coupled processor MIMD cluster of tightly coupled, shared memory, MIMD configuration of vector (SIMD) processors. The Clustered FORTRAN software support is described later.

III. *PARALLEL SOFTWARE*

IBM VS FORTRAN MULTITASKING FACILITY

In 1985, IBM announced a parallel interface extension to its VS FORTRAN compiler product. Although it consisted of only coarse-grained parallel features, it was still possible to perform useful work and to achieve parallel performance improvements. A description of this facility and performance results were presented at the 1986 workshop at ECMWF (ref. 7).

PARALLEL FORTRAN

Parallel FORTRAN is a facility for writing and executing parallel programs on IBM 3090 processors. It is based on the collaborative effort between IBM and Cornell Universities NSF center, where the prototype code was developed and tested. Two mechanisms were used to evaluate and gain experience with the compiler. The first was the strategic user program at Cornell University, where a broad spectrum of scientific investigators could obtain the benefit of parallel performance for their investigations, while evaluating the functions and features of the compiler. At the same time, IBM provided limited distribution of a version of this compiler as a special quotation (PRPQ) software (reference 10).

The Parallel FORTRAN compiler offers a number of parallel opportunities, as highlighted in figure 8. A robust set of parallel extensions are included to achieve both macro-scale and micro-scale parallelism. These extensions are summarized in figure 9, and include comment-based Directives which extend the existing vector Directives embodied in the underlying compiler product version. A sample parallel FORTRAN environment is shown in figure 10.

A brief summary of the parallel performance improvement using Parallel FORTRAN on an IBM 3090/600 with 6 vector facilities appears in table 5.

Application	Parallel Speed-up				Effective
	1	2	4	6	Parallel
Thin Layer CFD	1	1.8	3.0	3.9	89%
High Energy Physics QCD	1	1.8	3.1	4.3	92%
Statistical Methods	1	1.9	3.3	4.7	94%
Protein Folding	1	1.9	3.7	5.3	97%

Table 5: Parallel FORTRAN Performance

Figure 8: Highlights of Parallel FORTRAN

- Sharing of Storage Between Parallel Regions
- More Than One Level of Parallel Execution
- Dynamic Scheduling of Processors Throughout Execution
- I/O at All Levels of Parallel Execution
- Integrated Vector and Parallel Processing
- Parallel Execution Under Both MVS and VM

Figure 9: Parallel FORTRAN Extensions

LANGUAGE EXTENSIONS

- Parallel Tasks - Execute Subroutines in Parallel
- Parallel Cases - Execute Sections of Code in Parallel
- Parallel Loops - Execute Iterations of Loops in Parallel

COMPILER EXTENSIONS

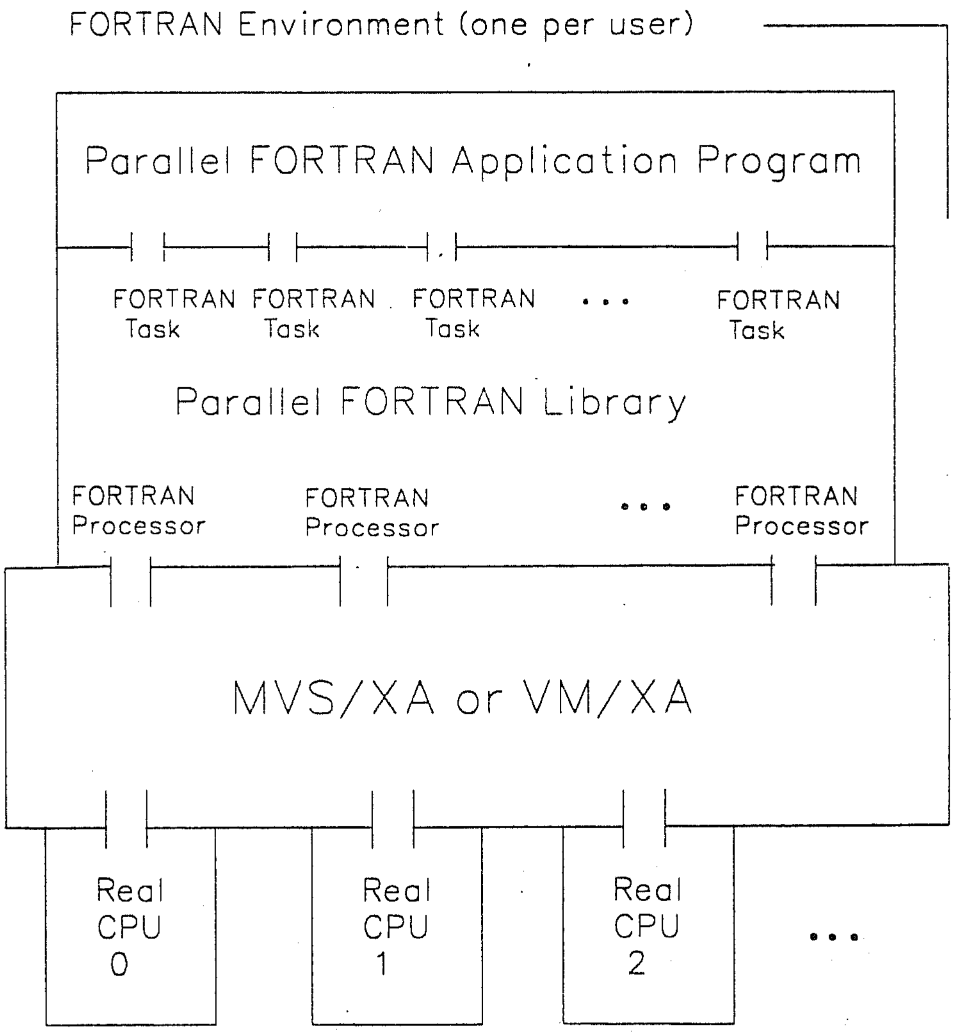
- Automatic Parallel - Execute Eligible DO Loops in Parallel

LIBRARY EXTENSIONS

- Library Functions - Locks, Events, Traces

The success of this collaborative activity has resulted in the adoption of the parallel features of Parallel FORTRAN into the IBM VS FORTRAN version 2, Release 5 program product. This compiler incorporates all of the scalar and vector features of the previous releases, including the multitasking. In fact, parallel opportunity which was considered to be too fine-grained for the multitasking (coarse-grained) approach was successfully exploited by the loop-level parallel capability of the compiler, achieving as much as 20% performance improvement through the use of directives alone, without other program modification.

Figure 10: Parallel FORTRAN Environment



CLUSTERED FORTRAN

The IBM-Cornell University activity has also evolved to include software to support the cluster of 3090/600 processors (12-way) now installed at Cornell as described earlier. Another set of extensions to the software and hardware have been defined. The IBM Clustered FORTRAN, a Supercomputer Systems Extension (SCSE) limited availability compiler and library, are based on IBM's VS FORTRAN version 2, release 3 compiler and library program product, including the functions and features of Parallel FORTRAN, described above. It also includes a node manager to manage multiple clustered FORTRAN address spaces, and a connection facility to handle data transmission between 3090 multiprocessors (MP's). The connection facility hardware is another SCSE and provides high speed coupling between two 3090 MP's. A diagram of the Clustered FORTRAN environment is shown in figure 11. This consists of a Root Virtual Computer (RVC) and Clustered Virtual Computers (CVC), plus associated support extensions (SCSE's). Clustered FORTRAN uses the IBM VM/XA operating system to manage and support the parallel environment. This software is described in reference 11.

LCAP/3090

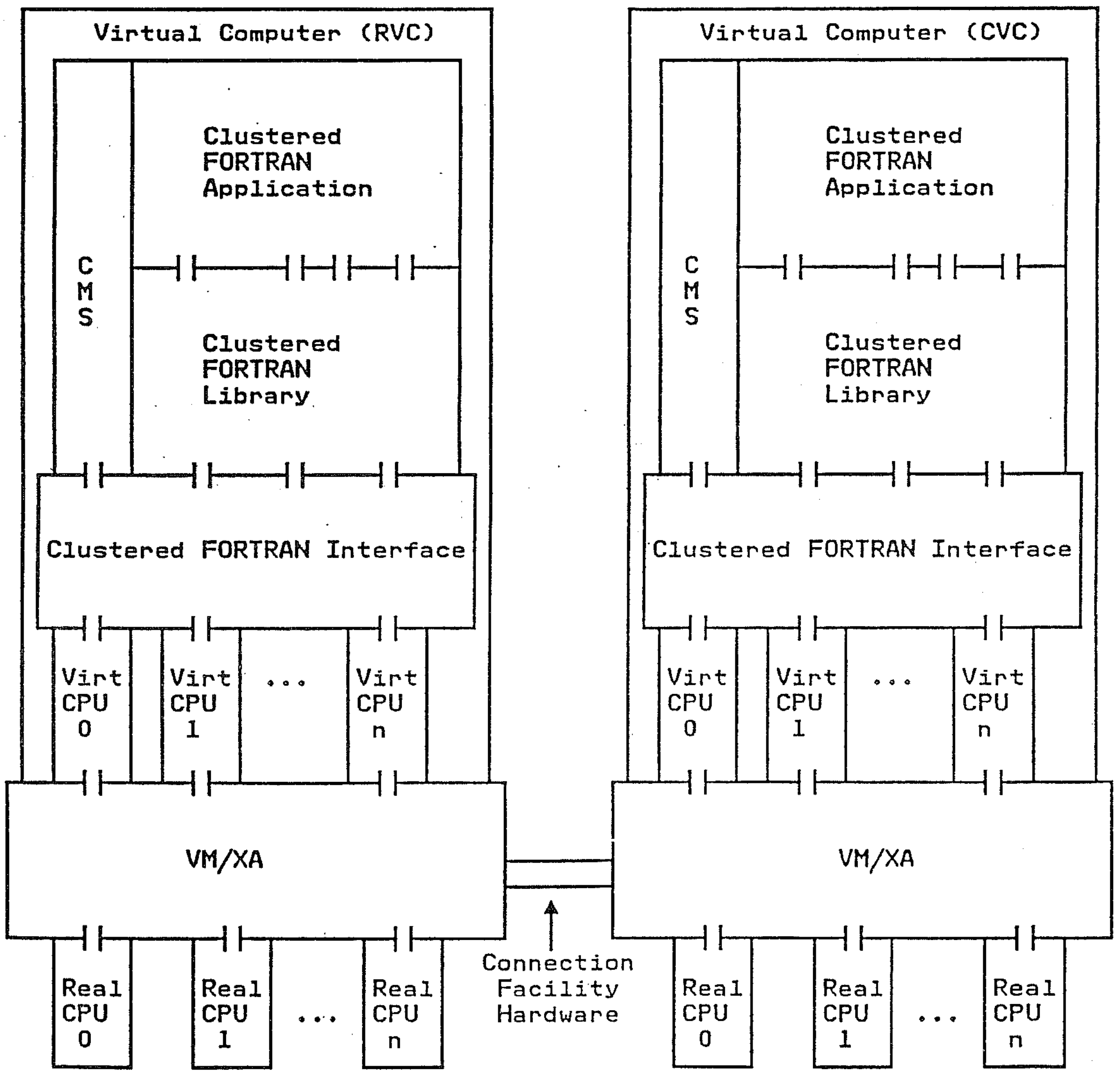
Another approach to clustered parallelism is found in the LCAP/3090 software. Originally developed to support a host IBM multiprocessor with numerous attached processors, the LCAP system was modified to provide access to clusters of IBM multiprocessors in an extendable manner. The basis for the coarse-grained parallelism which is the design paradigm for LCAP/3090 was discussed earlier. A complete documentation for LCAP/3090 can be found in ref. 12. This software system is an MIMD, master-slave oriented interface. The design philosophy of the software was to modify the underlying operating system as little as possible, and to build the LCAP/3090 system on top of IBM operating systems. In this way, the major portions of the software could be made operating-system independent and only particular pieces of the software would need modification in order to run LCAP on different operating systems.

LCAP/3090 uses the inherent support for parallel execution on IBM multiprocessors present in IBM operating systems. In the specific implementation of ref. 12, the IBM Virtual Machine, VM/XA system product is utilized. LCAP/3090 provides a sort of "shell" around VS FORTRAN in the form of:

Precompilers and Directives
Parallel Run Utilities
Run Scheduler and Resource Manager
Commands to Start, Monitor and Stop a Job
Communication Software

LCAP/3090 is operational at IBM's Kingston New York laboratory.

Figure 11: CLUSTERED FORTRAN ENVIRONMENT



IV. *CONCLUSIONS*

The parallel processing activities for scientific and engineering computing described here were initiated from a variety of perspectives and with a variety of objectives. Some were intended to investigate parallel machine architecture and design, some were aimed at the definition of parallel software interfaces, while still others had as a goal the solution of specific scientific problems, with parallel performance as the means. In all cases, the observation was made that the value of the machine was a strong function of the solution approach taken for a given problem, as expressed by application design, algorithm or data organization. No completely general design has been found which is equally effective across all of the many and variable aspects of parallel machine architecture, and design characteristics, and for all problem types and parallel granularity.

A number of approaches to machine architecture have been investigated, from SIMD to MIMD, tightly-coupled to loosely-coupled, shared memory to distributed memory. The balance between the number of processors and their power or complexity, the communication mechanism, switched or not, dynamic or static, the network topology and storage hierarchy, all in combination act to permit or prohibit the effective use of a particular application and application design on a particular parallel machine.

A number of examples of parallel processing advances in IBM have been discussed, including the RP3, GF11, VULCAN, LCAP and ES/9000 (3090) parallel machines, and IBM's Parallel FORTRAN, Clustered FORTRAN, LCAP/3090 and VS FORTRAN parallel software. Developed in a variety of research and collaborative efforts with universities and other research institutions, much of the detail and experiences of these efforts have been published in the open literature. Some of these results have been used to influence IBM products, and as the technology matures, been adopted into standard products themselves.

V. *REFERENCES*

1. "The IBM Research Parallel Processor Prototype (RP3)", Introduction and Architecture, G. Pfister, et al. Proceedings of the 12th International Conference on Parallel Processing

2. "The GF11 Parallel Computer", J. Beetem, M. Denneau, D. Weingarten IBM Technical Report TR 12364 Dec. 1986

3. "The GF11 Parallel Computer" - Programming and Performance M. Kumar, Y. Baransky IBM Research Report TR 15494 Feb. 1990

4. "An Implementation of Backpropagation Learning on GF11, A Large SIMD Parallel Computer", M. Whitbrock, M. Zagha "Parallel Computing" 14, North Holland, 1990

5. "LCAP/3090- Parallel Processing for Large-Scale Scientific and Engineering Problems", E. Clementi, D. Logan, J. Saarinen IBM Systems Journal Vol. 27, No. 4, 1988

6. "Using an IBM Multiprocessor System", A. L. Lim, D. B. Soll *Multiprocessing in Meteorological Models*, Ed. G-R Hoffmann, D. F. Snelling, Springer-Verlag 1988

7. "Parallel Processing on an IBM 3090 with Vector Facility", D. B. Soll *Multiprocessing in Meteorological Models*, Ed. G-R Hoffmann, D. F. Snelling, Springer-Verlag 1988

8. "Microtasking on IBM Multiprocessors", P. Carnevalli, P. Squazzero, V. Zecca, IBM Journal of Research and Development V. 30, No. 6 1986

9. *Parallel Computers 2*, R. W. Hockney, C. R. Jesshope Hilgar 1988

10. "IBM Parallel FORTRAN" - Language and Library Reference IBM Corporation SC23-0431-0 1988

11. "Parallel Processing on a Cluster of IBM ES/3090's", S. White IBM Corporation, May, 1990

12. *Modern Techniques in Computational Chemistry*, MOTECC (tm)-'90 - LCAP/3090, Ed. E. Clementi, Escom, 1990